

Graph Learning Methods for Genetic Interaction Networks

Dem Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
vorgelegte Dissertation

von
M.Sc. Fabio Nikolay
geboren am 30.09.1987 in Mainz am Rhein

Referent:	Prof. Dr.-Ing. Marius Pesavento
Korreferent:	Prof. Dr. Monica Bugallo
Tag der Einreichung:	25.06.2019
Tag der mündlichen Prüfung:	25.11.2019

Nikolay, Fabio: *Graph Learning Methods for Genetic Interaction Networks*

Darmstadt, Technische Universität Darmstadt

Jahr der Veröffentlichung der Dissertation auf TUpriints: 2019

Tag der mündlichen Prüfung: 25.11.2019

Veröffentlicht unter: CC BY-NC-ND 4.0 International

(<https://creativecommons.org/licenses/>)

URN: urn:nbn:de:tuda-tuprints-96343

URI: <https://tuprints.ulb.tu-darmstadt.de/id/eprint/9634>

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

30.11.2019, Fabio Nikolay

Datum und Unterschrift

Kurzfassung

Nicht nur die biomedizinische Forschung, sondern auch das Forschungsfeld der Genetik und die Genomforschung, befinden sich im Umbruch. Allen voran befindet sich die Genomforschung durch die rapide Entwicklung von *data sensing* und *data analysis*-Methoden auf dem Weg zu einer Daten getriebene Wissenschaft zu werden. Mit dem Aufkommen neuer *wet-lab* Technologien, wie der *synthetic genetic array* Technologie, sind große Mutationsexperimente an dem Genom von einfachen Organismen, wie zum Beispiel *E. coli*, schnell und kostengünstig durchführbar und sehr große Datenmengen verfügbar geworden. Aus den so gewonnenen Datenmengen neue Einblicke in grundlegende biologische Prozesse von Organismen zu erhalten, stellt eine beachtliche Herausforderung dar und wird in der Literature meist als *data-mining* oder *knowledge-engineering* bezeichnet. Einen wichtigen Aspekt der Genomforschung stellt die Untersuchung des Einflusses genetischer Interaktionen auf den Zellstoffwechsel dar. Folglich findet die Identifikation genetischer Interaktionen große Beachtung in der Wissenschaft. Die Interaktionen zwischen Genen einer bestimmten Menge können kompakt als genetisches Interaktionsnetzwerk dargestellt werden. Ein solches genetisches Interaktionsnetzwerk wird üblicherweise durch einen gerichteten Graphen dargestellt.

Aufgrund der Relevanz genetischer Interaktionen bezüglich des Zellstoffwechsels erfreut sich das Problem des Lernens solcher Interaktionsnetzwerke großer Aufmerksamkeit. Der Fokus dieser Dissertation liegt auf der Entwicklung von Graph-Lernverfahren, die explizit auf die spezielle Struktur der zu erlernenden genetischen Interaktionsgraphen angepasst sind. Der Hauptbeitrag dieser Arbeit besteht in der Formulierung des Graph-Lernproblems zur Identifikation genetischer Interaktionen als ganzzahlige Programme, wobei die den Daten zugrundeliegenden Interaktionsnetzwerke in verschiedenen Repräsentationsdomänen gelernt werden. Die auf ganzzahligen Programmen beruhenden Graph-Lernverfahren in dieser Dissertation bieten große Vorteile gegenüber generischen Verfahren zum Lernen von Graphstrukturen, wie zum Beispiel dem *Graphical Lasso*-Verfahren, da sie an die zugrundeliegenden biologischen Interaktionsmodelle angepasst sind. Diese Anpassung, der in dieser Arbeit vorgestellten Graph-Lernverfahren auf Basis diskreter Programme, an die zugrundeliegenden genetischen Interaktionsmodelle, erlaubt es, biologischen Gesetzmäßigkeiten in den geschätzten Interaktionsnetzwerken Rechnung zu tragen, welche mit generischen Graph-Lernverfahren nicht berücksichtigt werden können.

Aufgrund der kombinatorischen Natur der im Rahmen dieser Dissertation entwickelten, auf ganzzahligen Programmen beruhenden, Graph-Lernverfahren ist es nur unter Zuhilfenahme großer Computer-Cluster möglich, große genetische Interaktionsnetzwerke zu

berechnen. Daher wurde im Rahmen dieser Forschungsarbeit ein methodisches Framework entwickelt, welches basierend auf den vorgeschlagenen ganzzahligen Programmen zum Graphen-Lernen, in der Lage ist auch sehr große genetische Interaktionsnetzwerke approximativ zu lernen. Weiterhin wurde auch die Anwendbarkeit bekannter Verfahren des maschinellen Lernens auf das oben beschriebene Graph-Lernproblem untersucht.

Die Performanz der im Rahmen dieser Arbeit entwickelten Verfahren wurde hinsichtlich synthetischer und realer Daten untersucht und mit Verfahren vom Stand-der-Technik verglichen. Es wird gezeigt, dass die im Zuge dieser Dissertation entwickelten Verfahren den Methoden aus dem Stand-der-Technik, hinsichtlich synthetischer und realer Daten, überlegen sind. Abschließend werden die entwickelten, auf ganzzahligen Programmen beruhenden, Graph-Lernverfahren mit Methoden des maschinellen Lernens zum Graphenlernen verglichen. Es wird gezeigt, dass die Lernqualität der auf ganzzahligen Programmen beruhenden Graph-Lernverfahren höher ist, als die der Verfahren des maschinellen Lernens. Allerdings skalieren die untersuchten Verfahren des maschinellen Lernens wesentlich besser mit der Größe der zu schätzenden Netzwerke, als die entwickelten auf ganzzahligen Programmen beruhenden Verfahren.

Abstract

In recent years, the field of biomedical, genetics and genomics research is undergoing a major change. Especially genomics research is turning into a data-driven science, due to the rapid evolution of automatized data sensing technologies as well as data analysis methods. With the advent of new wet-lab technologies as the *synthetic genetic array* technique, the conduction of large-scale knockout experiments of thousands of genes has made a vast amount of data become available. With the knockout data at hand, a major challenge is the analysis of the available data which is commonly referred to as data-mining or knowledge-engineering. One important aspect of genomics research is the investigation of the influence of genetic interactions on the cell metabolism of organisms. For this purpose, the identification of genetic interactions with respect to a specific cell function is of high significance. Interactions among a collection of genes are well described by genetic interaction networks by means of directed graphs.

In light of the importance of understanding the influence of genetic interactions for the cell metabolism, the problem of learning genetic interaction networks, which reflect the mutual genetic dependencies among a set of genes, has recently attracted much attention. In this dissertation, the focus lies on developing graph learning algorithms dedicated to the special structure of genetic interaction networks. The main contribution of this work is the formulation of the graph learning problem as integer linear programs that estimate the genetic interaction network underlying the knockout data. In particular, the two proposed integer linear program based formulations are of different accents, since the network topology is estimated in different representation domains. The two methods have the advantage over conventional graph learning methods like *Graphical Lasso*, that the estimates of both proposed integer linear program based algorithms are guaranteed by design to have the desired network structure, which is imposed by the specific biological interaction model that is under study in this thesis.

Due to their intrinsic combinatorial nature, the proposed integer linear program based algorithms for graph learning cannot be applied to estimate large-scale genetic interaction networks. In order to compensate for this inability, a broader graph learning framework is presented which uses the proposed integer linear program based algorithms in an iterative fashion. Furthermore, “of-the-shelf” machine learning algorithms are customized to the graph learning problem.

The proposed integer linear program based methods are evaluated in terms of synthetic data as well as real data and compared to state-of-the-art methods. It is observed that the proposed integer linear program based algorithms are superior to the considered

state-of-the-art methods for both the synthetic data as well as the real data. Finally, the proposed integer linear program based algorithms are compared to selected “of-the-shelf” machine learning methods in terms of graph learning performance for synthetic data. Although the considered “of-the-shelf” machine learning methods cannot guarantee that their estimates are of the desired genetic interaction network structure, it is worth to mention that they yield a good tradeoff between estimation quality and the ability to estimate large-scale genetic interaction networks.

Contents

1	Introduction	1
1.1	Background	1
1.2	State-of-the-Art in Graph Learning	3
1.3	Thesis Overview and Contributions	15
2	Theoretical Background	19
2.1	Graph Terminology	19
2.2	Biological System Model	21
2.2.1	GI-Network Model	21
2.2.2	Data Model	23
2.2.3	Class Coupling	32
2.3	Annealed Importance Sampling	33
2.3.1	Simulated Annealing	34
2.3.2	Importance Sampling	38
2.3.3	Annealed Importance Sampling	40
2.4	Integer Programming	44
3	Integer Linear Programming for Graph Learning	51
3.1	GENIE Algorithm	52
3.1.1	Hierarchical Relationship Class Learning	52
3.1.2	Graph Topology Reconstruction	62
3.2	GI-GENIE Algorithm	72
3.2.1	ILP-Formulation of the GI-GENIE Algorithm	72
3.3	Scalability Techniques	81
3.4	Simulation Results	83
3.4.1	Synthetic Data Results	84
3.4.2	Real Data Results	106
3.5	Summary	116
4	Machine Learning Algorithms for Graph Learning	119
4.1	Feature Engineering	120
4.1.1	Hierarchical Relationship Class Learning	120
4.1.2	Direct DAG Topology Learning	123
4.2	Support Vector Machine Approach	124
4.3	Random Forest Classifier Approach	126
4.4	Artificial Neural Network Approach	129
4.5	Simulation Results	136

4.5.1	Hierarchical Relationship Class Learning	137
4.5.2	Direct DAG Topology Learning	150
4.6	Summary	166
5	Conclusions and Outlook	169
	Appendix	173
A.1	Appendix A	173
A.2	Appendix B - Full set of class coupling constraints \mathcal{L}	173
A.3	Appendix C - Full set of DAG-topology constraints \mathcal{L}_c	175
	List of Acronyms	179
	List of Symbols	181
	Bibliography	187
	List of Publications	199
	List of Supervised Students	201

Chapter 1

Introduction

1.1 Background

In recent years, there have been considerable advances in technology. Computer industry has made great progress in enhancing the performance of electronic devices in terms of computation power, energy efficiency and form factor of the devices [Che16], [Coo16], [Key00]. The development of computer hardware has been accompanied by a rapid evolution of intelligent algorithms that are increasingly able to perform cognitive tasks as for instance speech-and handwriting recognition, text generation/processing, autonomous driving and disease diagnosis [LBH15], [LJB⁺95], [Kri15], [Joa02], [WXL⁺14], [LZS16], [YJ17]. In addition to the above mentioned advances, the ability to produce and collect vast amounts of data in almost any branch of business, industry and social life, has proved to be the final part that makes intelligent algorithms deployable to real life problems. Together, the advances in computer hardware and designing intelligent algorithms, as well as the ability to generate and collect massive amounts of data, have been changing our social life as well as modern economy. New digital economic branches have been created and established economic branches such as automotive industry, banking and finance, healthcare and most services are being changed completely by intelligent machines/software. The way industry is manufacturing goods and products will be altered entirely by the deployment of automated robots and intelligent machines. Service industries are also facing an upheaval due to the increasingly intensive use of smart software and automated robots. The above mentioned changes for social life and economy are commonly referred to as *digital transformation*. An essential part of the *digital transformation* is the design of intelligent algorithms to solve “cognitive” problems/tasks. However, such cognitive tasks are very difficult to solve since there are no sequences of specific and well defined instructions known that have to be carried out to fulfill a cognitive task in general [Alp10]. In order to solve cognitive tasks as, for instance, the classification of emails to “spam”/“no-spam”, intelligent algorithms have to be designed that are not in need of a well defined sequence of instructions to solve the given task. Such algorithms have the ability to learn which steps to take to fulfill their task based on a sufficient amount of data which is representative for the problem at hand [Alp10]. The design of such intelligent algorithms is one of the main goals of machine learning (ML). Generally speaking, ML is the procedure of converting experience, i.e., observed data

of some task at hand, into expertise knowledge, i.e., conclusions about the task under study [SSBD14].

ML is an interdisciplinary field that shares common ground with the mathematical fields of statistics, information theory, game theory, optimization and graph theory [SSBD14] and can be divided into three main learning scenarios: supervised learning, unsupervised learning and reinforcement learning [SSBD14], [Alp10], [Mur12].

In supervised learning, the goal is to perform some task after having been trained on a set of data there is prior information on. The classification of spam email is a prominent example for supervised learning. Given a training set of emails that have been labeled spam/no-spam, a spam filter is learned on that training set in order to predict whether an unobserved email out of a test set of data is spam or not.

In unsupervised learning, there is no distinction between training and test dataset. Furthermore, there is only very limited prior information on the data set under study available. The objective in this learning scenario is to gain knowledge about the process/system underlying the given dataset. One common example is separating a dataset into several clusters. Clustering algorithms can be used to “label” data in the absence of human expert knowledge.

In reinforcement learning, the goal is to learn a scalar *value function* that models the reward of taking some action for interacting with the *environment* [SSBD14]. For instance, suppose the goal is to learn a value function that describes for each setting of a chess board the degree by which White’s position is better than Black’s. However, the only information available to the learner at training time are the positions that occurred throughout actual chess games, labeled by who has won that game [SSBD14].

Another important aspect of ML is the discipline of graph learning (GL) [Mur12], [NS12]. Depending on the learning setup, GL occurs in both the supervised and unsupervised learning scenario. In the past decade, GL has attracted much attention since many problems from various fields can be either reduced to learning the graph/network topology that is inherent to the problem, or they require some graph/network topology to be known. For instance, in modern medicine there is a strong increase in the use of decision-support-systems (DSS) [Tur93], [Ber07] in order to find the correct medical diagnosis for a huge variety of clinical symptoms. Typically, a DSS is implemented by a Bayesian network which is a statistical model that reflects the mutual dependencies among a set of random variables by an acyclic-directed-graph [J⁺96]. As a Bayesian network, a DSS is mathematically a network/graph of nodes that represent random variables, i.e., in the example of medical diagnosis the nodes are the different

symptoms/diseases. The topology of the Bayesian network, i.e., the graph/network topology, models the interactions of the nodes [Mur12], [J⁺96]. In order to operate a Bayesian network for decision support in the field of medical diagnosis, the interaction network of the various symptoms and diseases, i.e., the topology of the underlying Bayesian network, has to be learned. As another example for the significance of GL, the field of graph signal processing (GSP) has attracted much attention in recent years. In GSP, data and signals are modeled to originate from a graph structure that relates data/signal elements by dependency, similarity, physical proximity and other properties [OFK⁺17], [SM14]. GSP provides a toolbox to computationally efficiently analyze and process the constantly increasing amount of data that is being generated in many scientific and commercial domains. While GSP finds numerous applications in, for instance, biomedical and genetics research, fundamental physics, astronomical observations, social network analysis and consumer behaviour studies [SM14], learning the topology of the graph underlying the data poses a critical and fundamental challenge to the applicability of GSP. Especially in the field of bioinformatics and genomics, GL plays an important role. Experiments on various kinds of organisms are conducted with huge efforts to generate vast amounts of data that are used to gain knowledge about the complex metabolic processes and the functional organization of cells [BO04]. Based on the genomics data thus obtained, for instance, protein-protein interaction (PPI) networks and genetic interaction (GI) networks can be inferred [Tea18], [BO04], [SKS⁺11]. In particular, the network/graph topology, which is underlying the genomics data, has to be learned by adequate GL algorithms.

In this thesis, the main objective is the development of GL algorithms for the identification of genetic interaction networks based on various genomic data sources. The remainder of Chapter 1 is structured in two sections. In Section 1.2, an overview in state-of-the-art GL algorithms is provided. In Section 1.3, a detailed outline of this dissertation as well as an overview of the contributions is given.

1.2 State-of-the-Art in Graph Learning

Graphs are fundamental to model and represent complex dependencies among *objects* from various kinds of fields, where the nodes of the graphs represent the *objects* and the dependencies among the *objects* are reflected by the graph topology. For example, the *objects* can be random variables whose conditional dependencies are described by the network topology. Such graphical models of the joint probability distribution/density of multiple random variables are commonly referred to as Bayesian networks [Mur12], [J⁺96]. In geoinformatics, for instance, many applications are based

on data models that are described by graphs. As an example, modern navigation systems that are able to route vehicles or people from one location to another location are usually based on graph models, which describe the geography and the properties of the region where the start and the target locations are situated in. In this particular geoinformatical example, the *objects* model the locations while the physical connections between the locations and their associated distances are modeled by the graph topology. Irrespectively of the field of application and the scientific discipline, knowing the graph topology is essential. Rarely, there are situations where the graph topology of the application of interest and the scientific field, respectively, is known a priori or can be derived based on expert knowledge. However, in general the graph topologies are unknown and have to be learned based on data. Since graphs are a widespread concept to model complex dependencies, there are various types of graphs with different expressive power depending on the field of application and the branch of research they originate from. Furthermore, in each field of application and each branch of research, respectively, different methods have been developed in order to learn the network topology that is underlying the observations. In recent years, the development of GL algorithms has been mainly driven by five branches of research that are statistics, signal processing, GSP, ML and discrete optimization. Although being a subbranch of conventional signal processing, GSP bears several concepts and a special notion of the relationship between graphs and signals, which are exclusive to GSP and are not found in other signal processing disciplines. Therefore, in the interest of a better conceptual discriminability between the GL methods that stem from (overall) signal processing, the distinction into conventional signal processing and GSP is made. In each of the mentioned areas of research different GL methods for diverse purposes have been devised. In the following, the most prominent GL algorithms of the five mentioned fields of research are briefly revisited.

Statistics:

In statistics, GL algorithms have been designed for a plethora of purposes ranging from dependency and knowledge visualization to the estimation of large scale distributions in discrete, continuous and both discrete and continuous random variables [Mur12]. For instance, relevance graphs are often used in social media and biology to represent dependencies among random variables, [Mur12], [MNB⁺06]. In particular, a relevance graph visualizes the mutual information between random variables where an edge between two specific random variables is present if their mutual information is above a predefined threshold. Consequently, a relevance graph is composed of undirected edges. In the Gaussian case, the mutual information reduces to a logarithmic function of the correlation coefficient. Thus, in this case the relevance graph is also known as the covariance graph. Moreover, learning relevance graph topologies in this way assumes

that the decisions about the edges can be made independently of each other, i.e., the edges of the relevance graphs are estimated in a pairwise fashion. Although being appropriate for learning relevance networks, algorithms of pairwise edge estimation based on a specific measure, as for instance, the mutual information based pairwise thresholding, are of limited scope. This is due to the fact that for many structure learning problems it is known that the topology estimation is a joint problem in terms of edge estimation. Furthermore, the mutual information based learning method for relevance graphs tends to suffer from estimating overly dense graphs [Mur12].

An alternative of visualizing dependencies among a set of objects/random variables is given by dependency networks/graphs [Mur12]. A dependency network/graph of a set of random variables tends to reflect a sparse network topology of directed and undirected edges. In order to learn a dependency graph, full-conditional distributions based on Markov blankets are fit to the data using sparse regression and classification methods, respectively, [HCM⁺00], [MB06], [WRL06], [Dob07].

Furthermore, in statistics, graphs are used to model large joint probability-density-functions (PDFs) in an efficient way based on appropriate sets of conditional independence (CI) statements [Mur12]. Learning the dependencies/interactions among a set of random variables of a joint PDF for the purpose of knowledge discovery, estimation and prediction, translates to learning the corresponding set of CI statements that reflect a specific graph topology of the considered random variables. Hence, the problem of learning a joint PDF reduces to learning the corresponding graph structure. This problem has been shown to be NP hard for general graph topologies [Chi96]. For the case that the graph topology is restricted to be a tree, the problem of learning the corresponding joint PDF can be solved efficiently. The Chow-Liu algorithm [CL68] computes the maximum-likelihood tree structure that models the joint PDF based on Prim's algorithm and Kruskal's algorithm, respectively, [CT76], [Kru56]. For the case that the graph topology is restricted to be a directed-acyclic-graph (DAG), the problem of learning the corresponding joint PDF is called *Bayesian network structure learning* [Mur12]. However, there are fundamental limits to which the DAG topology can be learned. It has been shown that it is only possible to learn the DAG up to Markov equivalence, i.e., up to one unique set of CI-statements, [Mur12], [VP91]. However, a specific set of CI-statements encodes a set of DAGs with the same undirected skeleton and the same set of *v-structures* [VP91]. Thus, it is not possible to infer the correct orientations for all edges in the DAG. In order to learn the DAG that best describes the structure of the joint PDF under study the *K2-algorithm* can be used [CH92].

In some real life applications the assumption that the observed data is composed of all

random processes that contribute to the application under study is violated. Hence, there exist hidden/latent variables that contribute to the observed data which cannot be observed. In order to learn a DAG structure with latent variables for joint PDF estimation, sophisticated algorithms have been developed, as for instance, the Cheeseman-Stutz algorithm [CS96], the Variational Bayes EM algorithm [YL07] and the Structural EM algorithm [Fri97].

In some domains it is rather unusual to model dependencies among random variables by a directed graph structure, since there are no theoretical justifications for inducing a directional dependency between random variables [Mur12]. For instance, in image analysis, images are modeled by a 2D lattice graph structure where each node represents a pixel of the image and the edges between the nodes depict the pixel interaction in the image. As a common assumption in image processing, neighboring pixels are correlated, i.e., share edges with each other in the 2D lattice graph. Since the interaction among neighboring pixels is not of a hierarchical and causal nature, it is reasonable to consider the edges orientation free. Thus, the graph topology to be learned is undirected. Such a graph topology to model the interactions among a set of random variables is called *Markov Random Field* (MRF). Under the assumption that all random variables are Gaussian, a MRF is called *Gaussian Random Field* (GRF). In order to learn a GRF, that fact that there is a one-to-one correspondence between zeros in the precision matrix and the absence of edges in the GRF is exploited [Mur12]. The graph learning algorithm for GRF is commonly known as *Graphical Lasso* or *GLASSO* [Mur12]. The authors in [BG06], [BGd08], [DGK08] and [FHT08] have proposed efficient algorithms to solve the GLASSO problem. In order to learn the topology of a MRF with discrete random variables, the authors in [SMFR08] proposed an algorithm that is based on the idea of the GLASSO algorithm.

Conventional Signal Processing:

In [SBG17], the authors relate the observable graph data with the graph topology by a structural equation model (SEM). Resting on the SEM-based graph signal model, first a tensor factorization approach is proposed in [SBG17] to learn a directed and static graph structure that is underlying the data. As an extension to the GL method of inferring static directed graphs, the authors in [SBG17] propose a second GL algorithm, that is capable of tracking dynamic directed graph topologies, i.e., graphs that evolve over time.

In [GSK18], the authors provide a collection of selected graph data models along with corresponding algorithms to estimate the respective graph topologies. In order to identify graph topologies that model linear relationships within the nodal data that is collected over time, the authors in [GSK18] consider a Pearson correlation [Bri95]

based model, a SEM based model, a Granger causality based model as well as an vector-autoregressive model (VARM).

In particular, the Pearson correlation based approach to identify the graph structure, underlying the data, estimates an edge between two nodes in the graph if the Pearson correlation coefficient is above a pre-defined threshold. However, in this fashion unmediated effects cannot be separated from mediated effects, in the sense that an edge is estimated in the graph between two nodes if they are neighbors in the graph as well as if they are connected with each other via a cascade of other nodes. Hence, in this way overly dense network topologies are uncovered. To compensate for this shortcoming, the concept of partial correlations (PCs) is used to estimate graph topologies that reflect only unmediated, i.e., direct, effects among the nodes. Note that the classic Pearson correlation method as well as the PCs based method can only learn undirected graph topologies [GSK18]. A more complex representation of linear dependencies among nodal data is given by (linear) SEMs. Those models are composed of two kinds of (nodal) variables: endogenous and exogenous. The exogenous variables incorporate external effects into the network while the endogenous variables describe effects that are created by the network [Kap08]. The interactions among the two kinds of variables is captured by the underlying graph topology which can be estimated using a LS approach [CBG13]. In essence, SEM based data models for graph signals allow for topology estimates that can be both directed and undirected [BBG13]. Another popular approach for identifying directed networks is based on Granger causality [Gra69], where a directed edge between two nodes corresponds to a specified causal relationship between those nodes. The network topology is estimated according to a regression hypothesis test [Ham95]. Furthermore, a prominent choice of modelling the data generation and the inherent dependencies among nodal variables of complex systems is given by VARMs [CGS⁺11]. Although being similar to SEMs, VARMs are missing the distinction of variable types in the sense that there is only one type of variable which would be endogenous in the context of SEMs. In VARMs, the graph data at a specific point in time is not only based on the instantaneous nodal data and the corresponding generating graph topology, but also on time-lagged nodal data and their corresponding graph topologies. Hence, VARMs are able to capture temporal dynamics in the graph topology. The network structure, underlying the observed nodal data, can be estimated via an LS scheme [CGS⁺11].

Although being quite popular, the assumption that the nodal variables depend on each other only linearly falls to short for modelling some practical scenarios. Hence, there have been developed graph data models and corresponding GL algorithms that are able to capture non-linear dependencies among nodal variables [GSK18]. In particular, the (linear) PC method has been extended by non-linear kernel functions to account for

non-linear dependencies among the nodes in the graph as shown in [KGSL16], where the kernel parameter estimation can be reformulated as a convex program [ZRG16]. However, the non-linear PC based method presented can only infer undirected network topologies [KGSL16], [ZRG16]. Furthermore, the linear VARM of [CGS⁺11] is extended in order to capture non-linear relationships among the nodal variables. In particular, the linear vector autoregressive model equations, that describe the diffusion of nodal data across the network, are modified by a non-linear kernel function in such a way that non-linearly transformed nodal data is spread across the network over time. Finally, the resulting topology inference problem is solved using the ADMM method [BPC⁺11]. Similarly to their linear counterparts, non-linear VARMs are able to model both directed and undirected graph topologies which reflect non-linear dependencies. In essence, the kernel based VARM concept presented in [GSK18] can also take account of sparse and low rank adjacency matrices while scaling well with the network size and the amount of data available.

Recently, especially with the advent of social media and social networks, there has been an increasing interest in multi-layer network models, due to their ability to integrate information from various sources. For instance, assume that there is a set of individuals each participating in social networks, where the goal is to estimate the relationships among those individuals of interest. Due to their participation in social networks, each individual of interest generates a lot of data that can be interpreted as the data of the nodal variable which corresponds to the specific individual. Based on the social network specific nodal data of the individuals, the relationships among the nodes within that specific social network can be learned. If an individual is registered in multiple social networks, then the nodes corresponding to that particular individual in the different graph layers are connected. Hence, inter-layer connections are revealed by multi-layer network/graph models which could not be captured by single-layer network/graph models. The authors in [GSK18] propose a multi-layer network model of directed edges, that captures linear relationships among the nodes, based on an extension of the SEM. This extension is referred to as multi-layer SEM where the inference of the directed multi-layer network topology is formulated as a convex problem [TSG17] which is solved using the ADMM method [BPC⁺11], [SRG08].

Although being beyond the scope of this dissertation, graph data models with time-varying (TV) topologies are briefly recapitulate for the sake of completeness. In [GSK18], the authors review the most relevant models and methods for TV topology estimation. In particular, the authors in [ZLW10] propose an algorithm of estimating smoothly changing graph topologies, which are undirected and reflect linear dependencies, based on the well known GLASSO formulation. Furthermore, in [AG11] the underlying TV undirected graph topology is assumed to show switching behaviour in

the sense that the topology repeatedly remains constant for a certain time interval and then changes abruptly. In order to estimate the change/switching points and the corresponding undirected graph topologies the authors in [AG11] resort to a GLASSO based approach.

In order to model and estimate directed TV graph topologies, the authors in [BG17] propose a switched dynamic SEM approach where the graph topology is jumping across a pre-defined set of states which are related to different network structures. Note that the network topologies considered in [BG17] reflect linear dependencies among the nodal variables. This problem formulation results in a NP-hard mixed integer problem which is solved in an alternating manner [BG17]. As a major shortcoming in the context of TV topology estimation, the switched dynamic SEM approach is memory-less in the sense that the model equations do not involve time-lagged graph data and topology information. Hence, in order to overcome this shortcoming, VARM based methods for TV graph topology estimation have been investigated in [FSJW09] where the TV network topologies are assumed to stem from a discrete and finite state space. Furthermore, the states are assumed to follow a hierarchical Dirichlet process hidden Markov. Finally, the sequence of network topologies is estimated using Gibbs sampling. It is important to remark that the learned graph topologies in [FSJW09] reflect linear relationships among the nodal variables. Moreover, the dynamic VARM approach is able to model and learn both directed and undirected network topologies. As an alternative to the aforementioned models and methods for TV graph topology estimation, algorithms based on Bayesian networks are able to recover directed topologies with the interpretation that the edge directionality describes the causality [GSK18]. In [RH10], the likelihood rests on the Bayesian-Dirichlet equivalent metric where a Markov-Chain-Monte-Carlo (MCMC) sampler is used to sample from the posterior a sequence of graphs and the corresponding change points [GSK18].

Graph Signal Processing:

GSP has recently attracted much attention from various directions, since it provides a toolbox of new algorithms and concepts for data analysis and property/parameter inference [OFK⁺17], [SM14]. Contrary to data signals that are defined over regular domains, i.e., time or space, and which are encountered in, for instance, communications and control, GSP has come up with a new data paradigm which states that the observed data is related with the topology of a graph. In particular, the observed data is defined over the set of vertices of the underlying graph in the sense that each vertex of the graph is interpreted to generate a specific portion of the data. Such signals are commonly referred to as graph signals. The observed graph signals are assumed to originate from a hidden random process that is diffused over the graph

structure. In particular, the diffusion process, which creates the observed graph signal based on the hidden random process, is based on a graph-shift-operator (GSO) of the underlying graph [SMMR17], [SSMM17]. A GSO reflects direct and/or indirect relationships among the vertices of the underlying graph where common choices of the GSO are the adjacency matrix or the graph Laplacian matrix. Given the graph topology, the focus of GSP is to perform data analysis and parameter inference while exploiting the inherent structure of the data which is reflected by the underlying graph topology [MSLR16], [Han10], [CVSK15]. In practice, however, the graph topology is unknown or only inadequately known, respectively, and hence has to be learned based on the observed data before GSP analysis and inference can be done. In order to learn the graph topology based on the observed graph signals, two scenarios have to be considered: *i*) the observed graph signals are stationary and *ii*) the observed graph signals are non-stationary.

Under the assumption that the measured graph signals are stationary, i.e., the diffused hidden random process is white, the authors in [SMMR17] learn the corresponding symmetric GSO, which reflects the graph topology, based on the graph signal covariance matrix. Furthermore, the eigenvectors of the graph signal covariance matrix and the sought GSO are identical, due to the stationarity assumption. Note that the assumption of the GSO being symmetric implies that the corresponding graph is undirected. Furthermore, the estimation of the GSO, corresponding to the sought graph topology, is formulated as a convex optimization problem that entails several degrees of freedom in order to customize the GSO learning problem for different scenarios. Hence, the GL algorithm in [SMMR17] can be interpreted as a framework for network topology inference problems for GSOs that reflect undirected graphs. Additionally to single-graph estimation problems, there has recently been an increasing interest in learning multiple related graphs jointly based on subsets of the observed data [GLMZ11], [DWW14], [HS10]. In neuroscience, for instance, learning the brain functional networks of different patients based on their respective brain activity observations is a crucial task. In this scenario, the brain functional regions, i.e., the vertices of the graph in a mathematical context, are identical for all of the patients, whereas the connections among those regions, i.e., the graph topology, may differ from patient to patient [SWUM17]. Although being not identical, the brain functional networks of the patients are structurally close to each other. In GSP, the problem of learning multiple related graphs jointly based on subsets of the observed graph data is also addressed. In [SWUM17], a GL algorithm is proposed in order to learn the GSOs that are related to the graphs underlying the different subsets of graph data. Furthermore, the observed graph data is assumed to stem from a stationary diffusion processes in the different symmetric GSOs. The method proposed in [SWUM17] formulates the joint estimation of the related GSOs

as a convex optimization problem that is based on the eigenvectors of the covariance matrices of the available subsets of graph data. The algorithm in [SWUM17] of jointly learning multiple related GSOs is mainly based on the ideas in [SMMR17]. Under the assumptions that the observed graph data is stationary and the sought GSOs are symmetric, the methods presented in [SMMR17] and [SWUM17] represent very good tools of learning GSOs, i.e., the corresponding undirected network topologies. In some practical scenarios, the assumption that all eigenvectors of the graph signal covariance matrix are known, either perfectly or corrupted by noise, is not true [SMMR17]. In the context of GSP, this problem is commonly referred to as the network topology inference problem with incomplete spectral templates. In this case, the problem of learning the GSO underlying the data, i.e. the GL problem, is formulated as a convex optimization problem that settles on the known eigenvectors only, where the part of the GSO that corresponds to the unknown eigenspace of the graph signal covariance matrix is modeled as an additional optimization variable and estimated as well [SMMR17]. In summary, it is important to remark that under the assumption that the observed graph signals are stationary, the GSO is always symmetric, and hence, only undirected graph topologies can be learned.

In a more general scenario, the measured graph signals are assumed to be non-stationary, i.e., the diffused hidden random process has an arbitrary covariance matrix. The assumption of the hidden processes to be arbitrarily correlated is of a considerable relevance, since in many practical problems the hidden processes are colored. For instance, given that the hidden process models the opinion of individuals about a certain topic, which is diffused over a social network, then it is a reasonable assumption that the opinion of closely related individuals, i.e., individuals that are connected via the underlying social network, is correlated. In essence, the non-stationary assumption about the observed graph signals paves the way for much broader GL problems. Furthermore, the eigenvectors of the graph signal covariance matrix do not correspond any longer to the eigenvalues of the GSO, due to the covariance matrix of the hidden process which is no longer the identity matrix [SSMM17]. The graph signal covariance matrix is composed of the graph filter, which is a polynomial in the sought GSO, and the covariance matrix of the hidden process. Furthermore, the eigenvectors of the graph filter are identical to the ones of the sought GSO [SMMR17]. Hence, existing GL algorithms under the assumption that the graph signal is non-stationary leverage on estimating the graph filter, based on the graph signal covariance matrix first, and apply the GL framework of [SMMR17] afterwards in order to learn the GSO. Following the above mentioned two-stage procedure of first computing the eigenvectors of the graph signal covariance matrix and afterwards learning the GSO according to the methods in [SMMR17], [SWUM17], respectively, the authors in [SSMM17] propose GL/GSO-

learning algorithms for non-stationary graph signals with respect to different scenarios of the GSO based graph filters and the hidden random processes.

Assuming that the GSO based graph filter is symmetric and there are a plethora of *graph signal and hidden random process* samples available, the graph filter is estimated according to a least-squares (LS) formulation. Subsequently, based on the estimated graph filter the ideas presented in [SMMR17], [SWUM17] are applied to estimate the GSO. Furthermore, given that the GSO based graph filter is symmetric and only statistical information about the hidden random processes is available, i.e., the covariance matrix, the authors in [SSMM17] first propose a non-convex LS formulation, which is convexly reformulated as a semidefinite program (SDP) [Ma10], in order to estimate the graph filter underlying the observed data. Based on the learned graph filter, the GSO is estimated according to [SMMR17] again. Under the assumption that the GSO based graph filter is positive semidefinite and only the covariance matrix of the hidden random process is given, the graph filter can be estimated based on a LS formulation according to [SSMM17], where the GSO is again recovered according to the methods in [SMMR17]. It is important to remark that the GL algorithms mentioned so far, that deal with non-stationary graph signals, are only able to estimate GSOs that correspond to undirected graphs.

Finally, assuming that the GSO based graph filter is asymmetric and only the covariance matrix of the hidden random process is given, the authors in [SSMM18] propose a two-stage algorithm to learn the graph filter based on the non-stationary graph data. First, an *initial* graph filter is estimated based on a biconvex problem formulation that can be solved by the alternating LS algorithm [MVB18], or by the alternating direction method of multipliers (ADMM) [BPC⁺11]. In a second step, the so obtained initial graph filter is used to formulate the problem of finding the ultimate graph filter. The ultimate graph filter problem formulation involves a quadratic convex objective function and non-convex constraints. Hence, the graph filter found by the method of [SSMM18] is generally suboptimal. Finally, based on the estimated graph filter, the corresponding GSO is computed according to [SMMR17]. Note that the assumption in [SSMM18] of the GSO based graph filter being asymmetric implies that the estimated GSO reflects a directed graph topology.

Machine Learning:

In the field of ML there have been developed a plethora of methods for classification and regression that can be used for GL. Moreover, in the context of ML there have also evolved algorithms especially dedicated to learning network topologies in recent years. Since the problem of learning the topology of a graph essentially means to make a hard/soft decision about the presence or absence of an edge between each pair of

vertices in the graph, classification and regression algorithms can be utilized. There is a large collection of such ML algorithms, as for instance, linear-and nonlinear regression algorithms [DS98], logistic regression methods [Fre09], [Fre09], support-vector-machines (SVM)s [CST00], [CV95], [Vap00] regression-and classification tree/forest algorithms [Bre01] and artificial neural networks (ANN)s [Bis95], [GBC16].

Discrete Optimization:

Apart from GL algorithms that originate from the fields of research mentioned above, there are a plethora of GL algorithms based on discrete optimization methods. In particular, learning the graph topology underlying the observed data is formulated as a integer linear program (ILP) that can be solved efficiently by *branch-and-bound* and *branch-and-cut* algorithms, respectively [LW66], [PR91], [AKM05]. One of the most intensively studied ILP is the *traveling-salesman-problem* (TSP) of Dantzig et al. [DFJ54], [CBD11], [PW06]. For the TSP, a salesman is forced to visit a collection of cities in such an order that each city is visited at most once and the total traveling distance is minimal. As a combinatorial problem, the TSP has been shown to translate to a graph learning problem [CBD11], [PW06]. The cities to be visited are modeled by a set of nodes and the distance between each pair of cities is described by a weighted edge between the nodes that correspond to that pair of cities. Given a starting point/city, the solution to the TSP is a graph that contains all cities as its nodes and a set of edges such that *traveling* along the direction of those edges leads to the starting point/city with visiting any city at most once [DFJ54], [CBD11], [PW06]. In essence, the solution to the TSP is a *Hamiltonian cycle* of the shortest length [CBD11]. There are many variants of the TSP, as for instance, the *multiple traveling-salesmen-problem* (MTSP), the clustered TSP and the generalized TSP [CBD11]. Furthermore, many combinatorial problems from numerous fields can be reformulated as a TSP or one of its variants [CBD11]. In general, the formulation of GL problems as ILP paves the way for highly customized GL algorithms that show strong performance in terms of learning accuracy and which are able to ensure a pre-defined structure of the solution. However, the strong performance in terms of learning accuracy of ILP based GL algorithms comes at the cost of a poor scalability due to the combinatorial nature of the ILP formulation.

As a way of solving continuous and discrete optimization problems approximately, MCMC based methods have established as a popular alternative to ILP based methods for GL. Especially, the simulated annealing (SA) algorithm [KGV83], as a state-of-the-art MCMC algorithm to solve combinatorial problems, has shown to be a good meta-heuristic alternative for GL problems, and is oftenly used to compute solutions to the famous TSP [ZLZZ16], [LBL16], [CSMH91]. In SA, the original discrete optimization problem, as which the GL problem is formulated, is translated into a distribution over

the feasible set of the original problem. Based on a specifically constructed Markov-Chain, samples are drawn from this distribution where each drawn sample represents a candidate solution to the original problem. Finally, an approximate solution to the original discrete optimization problem, i.e., the GL problem, is obtained as the drawn sample which yields the smallest objective value (in an minimization context). There are several variants of the SA algorithm in order to find solutions to GL problems, as for instance, the annealed importance sampling (AIS) [Nea01] algorithm, which combines the ideas of SA and importance sampling (IS) [GI89], [RK16]. However, the SA algorithm and its variants are only able to yield sub-optimal solutions to the GL problem, due to their metaheuristic nature. Furthermore, those methods are widely known to be slow in terms of finding good approximate solutions to the original GL problem, since a plethora of samples have to be drawn in order to make the Markov-Chain converge.

GL Method Assessment:

The goal of this thesis is to develop GL algorithms for estimating (time)-static interaction networks of genes of interest based on the biological data model of [BJW⁺10]. The data model of [BJW⁺10] assumes that the dependencies among the genes under study are of a directed linear nature, i.e., the considered genetic interaction networks (GI-networks) are modeled by a directed graph of a special topology whose edges encode linear dependencies. The model relates the data, which originates from single genes and all pairs of genes, to hierarchical information about the graph topology that is encoded by a corresponding set of hierarchical interaction classes. In particular, for each pair of genes the observed genetic data is translated to a specific hierarchical interaction class. It is important to remark that the gene pairs under study cannot be translated to hierarchical interaction classes independently of each other, since they are coupled via the GI-network topology. Furthermore, the data is assumed to be time-static in the sense that the data remains static over time.

Given the topology of the underlying GI-network, the considered data model cannot be represented as a hidden process that is diffused over the network over time. Hence, the outlined GSP based GL algorithms cannot be applied for the purpose of learning GI-networks according to the data model of [BJW⁺10]. Furthermore, the Pearson correlation and PC based methods, that enjoy great popularity in the signal processing community, cannot be used in order to learn GI-networks based on the data model of [BJW⁺10]. They only allow for estimating undirected graph topologies and cannot account for the special structure of the considered biological data model. Although being applicable to a vast variety of structure learning problems, the outlined SEM based approaches as well as the VARMA based approaches are not suitable for the topology learning problem under investigation in this thesis. For SEMs and VARMA, respectively, the observed data is assumed to originate from nodal variables that are

connected with each other by the graph structure. In the case of SEMs, the nodal data at a particular node is assumed to be composed of the data of the neighboring nodes as well as of an exogenous nodal variable. In the case of VARMs, the nodal data at a particular node is assumed to be composed of the data of the neighboring nodes and their time-lagged versions. However, those model assumption are in conflict with the model of [BJW⁺10]. In statistics, there have been developed a variety of GL methods for the purpose of learning Bayesian networks that reflect large PDFs. In principal, being able to estimate directed graph topologies, such GL methods suffer from the shortcoming that the edge orientations cannot be identified uniquely. Furthermore, the GL algorithms outlined in the context of statistics, GSP and signal processing are not able to ensure the special structural requirements that are posed to the underlying GI-networks by the data model of [BJW⁺10].

On the other hand, discrete optimization based methods allow for customized modelling of arbitrary topology constraints. Hence, the GL algorithms proposed in this work, are mainly based in discrete optimization based approaches. Additionally, in order to obtain a good tradeoff in terms of topology estimation quality and scalability with respect to the number of genes under study, classical ML methods for classification are utilized to learn the GI-network topologies approximately.

1.3 Thesis Overview and Contributions

The goal of this dissertation is to study and develop GL algorithms for the purpose of estimating GI-networks based on large-scale single knockout (SK) data and double knockout data (DK). In particular, the emphasis of this thesis lies on the development of customized GL algorithms for GI-network reconstruction based on ILP formulations. Additionally, in order to overcome the considerable computational costs of the proposed ILP based GL algorithms for GI-network reconstruction, algorithms that are based on well known ML methods for classification and regression, i.e., SVM, random forest classifier (RFC) and ANN, are proposed. These methods provide a considerably good tradeoff between computational complexity and GI-network reconstruction accuracy. The algorithms proposed in this thesis represent valuable tools for computational biologists who attempt to gain deep knowledge about the cell metabolism and its regulation by genetic interactions.

The detailed outline of this dissertation is as follows:

In **Chapter 2**, the theoretical background and the biological system model, that is referred to throughout this thesis, is presented. First, prerequisites in graph theory

comprising the mathematical description and fundamental concepts are introduced. Following the introduction of the selected prerequisites in graph theory, a graph theoretic description of general GI-networks is presented and briefly elucidated. In the sequel, GI-networks according to [BJW⁺10] are introduced which represent a specific type of GI-networks. In particular, the specific structure of GI-networks under the terms of [BJW⁺10] is discussed in detail along with its biological meaning. Furthermore, the data model corresponding to the GI-networks according to [BJW⁺10] is presented in detail and further illustrated on the basis of simple examples. Throughout this thesis, only GI-networks under the terms of [BJW⁺10] are considered. Moreover, arising problems in learning the considered GI-networks are pointed out and explained in more detail by the help of two basic examples. As a prominent state-of-the-art method for solving discrete optimization problems, especially in the context of graph learning, the concept of annealed importance sampling is presented which is a popular adaptation of the well known simulated annealing algorithm. Finally, the topic of integer programming is briefly touched in order to take account for the ILP-based graph learning algorithms which constitute the focus of this work.

In **Chapter 3**, the ILP-based graph learning algorithms, i.e., the *genetic interactions detector* (GENIE) algorithm and the *genetic interactions-profile data extended genetic interactions detector* (GI-GENIE) algorithm, are presented. In particular, the GENIE algorithm learns the topology of GI-networks according to [BJW⁺10] in a two-stage procedure based on SK/DK data only. On the contrary, the GI-GENIE algorithm learns the topology of the considered types of GI-networks directly based on multiple data types. In this context, the GENIE and the GI-GENIE algorithms explicitly make use of the well known biological interaction model of [BJW⁺10] which relates single knockout and double knockout measurements to hierarchical dependencies in a GI-network. Due to the combinatorial nature of the proposed ILP formulation based algorithms, the GENIE and the GI-GENIE algorithm, respectively, are embedded into a broader framework denoted as *sequential scalability technique* (SEQSCA) in order to be able to process the data of large scale knockout experiments. Finally, the performance of the proposed GENIE and GI-GENIE algorithms is evaluated in terms of synthetic data as well as real data.

In **Chapter 4**, well known ML algorithms, i.e., the multiclass SVM, the RFC and the ANN, are briefly explained and utilized to learn the topology of the considered GI-networks. In particular, differently parameterized multiclass SVMs, RFCs and ANNs are trained based on two different feature design approaches both for synthetic data. Finally, the graph learning performance of the considered ML algorithms is evaluated with respect to synthetic data and compared to the performance of the ILP formulation based algorithms, i.e., the GENIE and the GI-GENIE algorithms, respectively.

Finally, in **Chapter 5** conclusions are drawn and future work is discussed.

Chapter 2

Theoretical Background

In this chapter, background information in graph theory, the biological system model for GI-network estimation, a prominent metaheuristic for estimating solutions of integer programs and basic concepts of integer programming are revisited.

In particular, in Section 2.1 fundamental graph theoretic principles along with the common terminology is introduced.

In Section 2.2 a graph theoretic description of a GI-network is presented in company with an illustration of the biological meaning. Following the graph theory based description of GI-networks, the biological system model of [BJW⁺10] is presented. The model of [BJW⁺10] provides a description to classify all gene pairs in a GI-network into biologically justified interaction types as well as the corresponding data representation. Furthermore, fundamental problems that arise in reconstructing the GI-network based on the biological model of [BJW⁺10] are discussed.

Furthermore, in Section 2.3, the *annealed importance sampling* algorithm is presented. As a prominent metaheuristic, the AIS algorithm combines the principles of *importance sampling* and *simulated annealing* in order to provide approximate solutions to integer and mixed-integer problems (IPs/MIPs).

In Section 2.4 optimization methods for integer problems are briefly discussed. A plethora of problems from a wide range of applications, for instance, the GI-network reconstruction problem, genome sequencing, process scheduling, etc., can be formulated as integer optimization problems. Therefore, algorithms to solve discrete problems constitute a key technology to many applications of today's emerging digital economy.

2.1 Graph Terminology

According to [Die06], a graph $\mathcal{A} = (V(\mathcal{A}), E(\mathcal{A}))$ is well defined by a set of nodes

$$V(\mathcal{A}) = \{a_1, a_2, \dots, a_A\}, \quad (2.1)$$

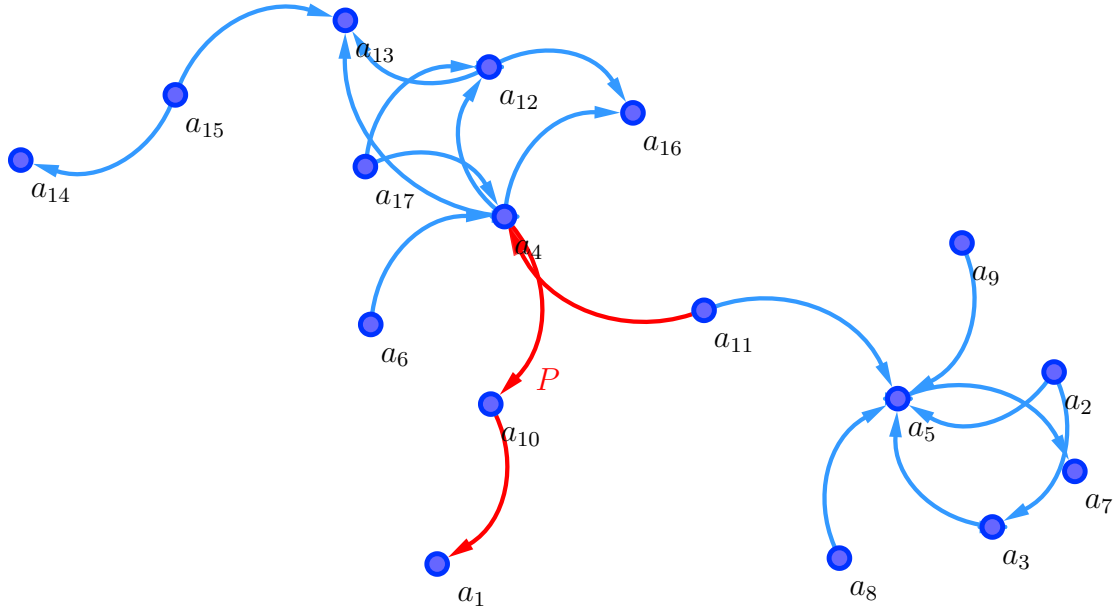


Figure 2.1. Example graph \mathcal{A}_1 consisting of vertices $V_{\mathcal{A}} = \{a_1, \dots, a_{17}\}$ and set of edges $E_{\mathcal{A}}$ that reflects the displayed topology. The edges highlighted in red constitute path P .

where cardinality $A = |V(\mathcal{A})|$ denotes the number of vertices in graph \mathcal{A} , and a set of edges

$$E(\mathcal{A}) = \{\{a_i, a_j\} \mid \{a_i, a_j\} \in \mathcal{A}\}. \quad (2.2)$$

which contains all edges $\{a_i, a_j\}$ connecting node i with node j in \mathcal{A} . On a coarse scale graphs can be distinguished into directed graphs and undirected graphs. In case of a directed graph, the edge $\{a_i, a_j\}$ linking vertices a_i and a_j is orientated from node a_i to a_j . In contrast to that, an edge $\{a_i, a_j\}$ in an undirected graph has no direction and thus, $\{a_i, a_j\} = \{a_j, a_i\}$. The operators $V(\cdot)$ and $E(\cdot)$ applied to graph \mathcal{A} yield the set of nodes $V(\mathcal{A})$ and the set of edges $E(\mathcal{A})$ respectively. For notational convenience, we mostly address the set of vertices $V(\mathcal{A})$ and the set of edges $E(\mathcal{A})$ of graph \mathcal{A} by $\mathcal{V}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{A}}$, respectively. A very important concept in graph theory, particularly in the context of this dissertation, are paths. According to [Die06], a path \mathcal{P} in some graph \mathcal{A} is a non-empty graph with set of vertices $\mathcal{V}_{\mathcal{P}} = \{a_0, a_1, \dots, a_{\gamma}\}$ and set of edges $\mathcal{E}_{\mathcal{P}} = \{\{a_0, a_1\}, \{a_1, a_2\}, \dots, \{a_{\gamma-1}, a_{\gamma}\}\}$ where all the vertices in $\mathcal{V}_{\mathcal{P}}$ are distinct. Path \mathcal{P} links the vertices a_0 and a_{γ} in graph \mathcal{A} . For notational convenience, path \mathcal{P} is mostly referred to as the sequence of vertices P that are traversed while travelling from vertex a_0 to vertex a_{γ} along the edges in $\mathcal{E}_{\mathcal{P}}$. Formally, path P from vertex/node a_0 to a_{γ} is described by

$$P = a_0, \dots, a_{\gamma} \quad (2.3)$$

where $\mathcal{V}_P = V(P) = \{a_0, \dots, a_\gamma\}$ denotes the set of nodes that are traversed by path P [Die06].

Example 2.1.1. *As highlighted in red in Fig. 2.1, path P describes a way from node a_{11} to node a_1 in graph \mathcal{A}_1 . Travelling along path P starting with node a_{11} , nodes a_4 , a_{10} and a_1 are visited in the stated order. Mathematically according to Eq. (2.3), path P in graph \mathcal{A}_1 in Fig. 2.1 is described by:*

$$P = a_{11}, a_4, a_{10}, a_1 \quad (2.4)$$

Another important concept to describe the topology of a graph is the adjacency matrix. Given graph $\mathcal{A} = (V(\mathcal{A}), E(\mathcal{A}))$, the adjacency matrix \mathbf{A} of graph \mathcal{A} is defined as $\mathbf{A} \in \{0, 1\}^{A \times A}$ where each element $\mathbf{A}(i, j) = a_{i,j} \forall a_i, a_j \in \mathcal{V}_\mathcal{A}$ is given by [Die06]:

$$\mathbf{A}(i, j) = a_{i,j} = \begin{cases} 1 & \text{if } \{a_i, a_j\} \in \mathcal{E}_\mathcal{A} \\ 0 & \text{otherwise} \end{cases} \quad (2.5)$$

Note that for undirected graphs, the adjacency matrix is symmetric whereas for directed graphs it is not.

2.2 Biological System Model

2.2.1 GI-Network Model

In the field of genomics research, gene-gene interactions with respect to a specific cell function under study are modeled as a GI-network [DCB⁺09], [C⁺10], [D⁺05]. In particular, a GI-network \mathcal{GI} that describes the interactions among a set of genes $\mathcal{V}_{\mathcal{GI}} = \{g_1, \dots, g_G\}$, with $G = |\mathcal{V}_{\mathcal{GI}}|$ denoting the number of genes in GI-network \mathcal{GI} , is modeled as a graph. According to Section 2.1, GI-network \mathcal{GI} is characterized by

$$\mathcal{GI} = (\mathcal{V}_{\mathcal{GI}}, \mathcal{E}_{\mathcal{GI}}) \quad (2.6)$$

where $\mathcal{V}_{\mathcal{GI}}$ denotes the set of genes under study and $\mathcal{E}_{\mathcal{GI}}$ denotes the set of edges of GI-network \mathcal{GI} . In general, the interactions among the investigated genes is reflected by the topology of the GI-network \mathcal{GI} , i.e., by $\mathcal{E}_{\mathcal{GI}}$.

In genomics, there are various gene-gene interaction models that are motivated by different biological assumptions. GI-networks based on different interaction models have different types of edges with different biological meanings, for instance, undirected

edges/interactions, directed edges/interactions and multiple edge types to describe more detailed biological relationships between genes. In addition to that, GI-networks of different interaction models have varying assumptions on the network topology. In this dissertation, the proposed algorithms and techniques to estimate GI-networks are based on the genetic interaction model of [BJW⁺10]. In this model, a GI-network is characterized by a DAG that is connected by a common root node called the *reporter level* R . All edges in such a GI-network are orientated towards the root node R , i.e., each path in a DAG under the terms of [BJW⁺10] finally terminates in the reporter node R and any gene appears on that path at most once. Hence, there exist no cycles. The report level R is an artificial node in the concept of a GI-network according to [BJW⁺10], i.e., it is not a gene, and represents the measured *phenotype* of the specific cell process under study. In Biology, the term phenotype describes the particular manifestation of a biological attribute of an organism that can be observed. For instance, a common biological attribute of bacteria is growth measured in colony size, where a particular size of the bacteria colony is a phenotype of this biological attribute. As another example, the hair color of human beings is a biological attribute of the human species where a specific hair color, for instance, red hair, is a phenotype of that attribute. The GI-network after [BJW⁺10] reflects the role of the studied genes for the execution of a specific metabolic process in the cell machinery in a hierarchical sense. Furthermore, the topological relationship between a pair of genes in such a GI-network describes their genetic interaction type. In summary, a GI-network under the terms of [BJW⁺10] encodes hierarchical information about the functional interplay of genes and helps Computational Biologists to gain valuable insights into the very basic entities of the human body: the cells. It is important to remark that in the model of [BJW⁺10] a GI-network only contains hierarchical information and not quantitative information about the actual observable effects of the interactions. Due to the described assumptions of the interaction model, GI-networks/DAGs according to [BJW⁺10] are denoted by \mathcal{D} and are characterized in a similar fashion as the general GI-networks as described in Eq. (2.6) by

$$\mathcal{D} = (\mathcal{V}_{\mathcal{D}}, \mathcal{E}_{\mathcal{D}}) = \left(\mathcal{G} \cup \{R\}, \mathcal{E}_{\mathcal{D}} \right) \quad (2.7)$$

where the set of vertices $\mathcal{V}_{\mathcal{D}}$ of DAG \mathcal{D} is composed of the set of genes under study, i.e., \mathcal{G} , and the reporter node R . In order to provide a better understanding about GI-networks/DAGs according to the model of [BJW⁺10], the following two examples give an aid to interpretation based on the GI-network displayed in Figure 2.2.

Example 2.2.1. In DAG \mathcal{D}_0 as displayed in Figure 2.2 there is an edge from gene g_0 to gene g_1 , i.e., $\{g_0, g_1\} \in \mathcal{E}_{\mathcal{D}_0}$, which indicates that the activity of gene g_0 controls the activity of gene g_1 . Hence, gene g_0 only affects the phenotype of the cell function

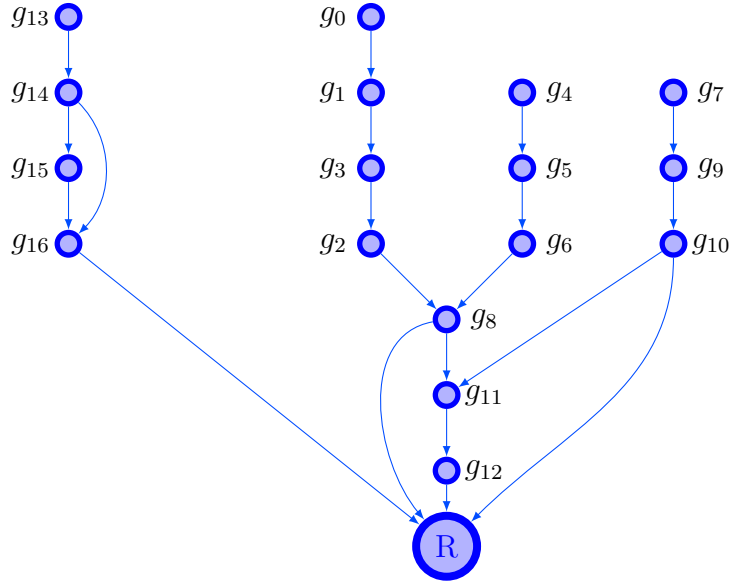


Figure 2.2. Example DAG \mathcal{D}_0 consisting of vertices/genes $\mathcal{V}_{\mathcal{D}_0} = \{g_0, \dots, g_{16}, R\}$ and a set of edges $\mathcal{E}_{\mathcal{D}_0}$ that reflects the displayed topology.

under study via gene g_1 and not directly. Note that the edge $\{g_0, g_1\}$ does not encode any quantitative information about the interaction of the genes g_0 and g_1 , respectively. Hence, the edge $\{g_0, g_1\}$ does not state if a strong/weak activity of gene g_0 affects the activity of gene g_1 in a positiv/negative way.

Example 2.2.2. In DAG \mathcal{D}_0 as displayed in Figure 2.2 there is no direct edge from gene g_0 to gene g_2 , i.e., $\{g_0, g_2\} \notin \mathcal{E}_{\mathcal{D}_0}$. Hence, there is no direct connection between the activity of gene g_0 and gene g_2 . However, with respect to the root node R genes g_0 and g_2 are in the same path/pathway and hence, they are connected indirectly. Thus, via a cascade of other genes the activity of gene g_0 indirectly affects the activity of gene g_2 .

Note that for the sake of notational convenience genes will be mostly addressed by their index in the following, i.e., gene g_i will be addressed by its index i .

2.2.2 Data Model

Given a set of genes $\mathcal{G} = \{g_1, \dots, g_G\}$ and a cell function under study. In order to quantify the influence of a gene $i := g_i \in \mathcal{G}$ on the specified cell function, gene i is functionally disabled, i.e., it is not able to participate in the metabolic processes of the cell, and the resulting phenotype is measured. In this case, gene i is considered to

be *knocked out* and the resulting phenotype, denoted as $R(i) \in \mathbb{R}$, is called the *single knockout* (SK) phenotype of gene i . Consequently, a *double knockout* (DK) $R(i, j) \in \mathbb{R}$ denotes the measured phenotype given that genes $i, j \in \mathcal{G}$ are functionally disabled, i.e., knocked out.

In a more formal way, SKs, DKs as well as higher order knockouts can be defined as $R(\mathcal{S}_b) \in \mathbb{R}$ that is the measured phenotypes of a subset of genes $\mathcal{S}_b = \{i_1, \dots, i_{\tilde{n}}, \dots, i_n\} \subset \mathcal{G}$, $1 < n \leq G$, $i_1 \neq i_{\tilde{n}} \forall i_{\tilde{n}} \in \mathcal{S}_b \setminus \{i_1\}$, being knocked out. In genomics research, there are a plethora of data models for the purpose of relating knockout measurements as for instance, SK, DKs, higher order knockouts and other knockout measurement based data types, to different kinds of GI-network topology information. Given a set of genes \mathcal{G} and the corresponding collection of knockout measurements of all possible orders, i.e., $\mathcal{R} = \{\mathcal{R}_{\mathcal{S}_{b,1}}, \dots, \mathcal{R}_{\mathcal{S}_{b,n}}, \dots, \mathcal{R}_{\mathcal{S}_{b,G}}\}$ where $\mathcal{R}_{\mathcal{S}_{b,n}}$ denotes the collection of knockouts of all possible subsets $\mathcal{S}_b = \{i_1, \dots, i_{\tilde{n}}, \dots, i_n\}$ of cardinality n . Moreover, let $\mathcal{C} = \{\Psi_1(\mathcal{R}), \dots, \Psi_{\tilde{n}}(\mathcal{R}), \dots, \Psi_{n_\Psi}(\mathcal{R})\}$, where $n_\Psi \geq 1$, denote the set of all possible data types based on knockout observations, with transformation functions $\Psi_1(\cdot)$ to $\Psi_{n_\Psi}(\cdot)$.

A model \mathfrak{M} relates an $n_{\mathfrak{M}}$ -dimensional tuple $\mathbf{t}_{\mathfrak{M}}$ of knockout measurements of different orders, as well as data types derived from knockout measurements, to GI-network topology information $T_{\mathfrak{M}}$ as depicted by Eq. (2.8):

$$\mathfrak{M} : \mathbf{t}_{\mathfrak{M}} \mapsto T_{\mathfrak{M}}. \quad (2.8)$$

The $n_{\mathfrak{M}}$ -dimensional tuple $\mathbf{t}_{\mathfrak{M}}$ is given by

$$\mathbf{t}_{\mathfrak{M}} = (t_{\mathfrak{M},1}, \dots, t_{\mathfrak{M},\tilde{n}_{\mathfrak{M}}}, \dots, t_{\mathfrak{M},n_{\mathfrak{M}}}) \quad (2.9)$$

where each $t_{\mathfrak{M},\tilde{n}_{\mathfrak{M}}} \in \mathcal{R}$ or $t_{\mathfrak{M},\tilde{n}_{\mathfrak{M}}} \in \mathcal{C}$, with $t_{\mathfrak{M},1} \neq t_{\mathfrak{M},\tilde{n}_{\mathfrak{M}}} \forall t_{\mathfrak{M},\tilde{n}_{\mathfrak{M}}} \in \mathbf{t}_{\mathfrak{M}} \setminus \{t_{\mathfrak{M},1}\}$.

As an abstract description of mutual gene-gene dependencies of a general form in the context of a GI-network, the GI-network topology information $T_{\mathfrak{M}}$ reflects various kinds of topology and interaction information, depending on the model \mathfrak{M} . For instance, $T_{\mathfrak{M}}$ could reflect:

- a set of directed / undirected edges $\mathcal{E}_{\mathcal{GI}}$ of multiple types of edges / solitary type of edges which describes the potentially various genetic dependencies in the underlying GI-network
- a representation of a GI-network \mathcal{GI} of potentially various genetic interaction types in a different domain, e.g., as a set of interaction classes that encode topology information of the underlying GI-network as well as potentially various genetic interaction types

- a representation of a GI-network \mathcal{GI} in a compressed form

In this thesis, only the model of [BJW⁺10] is considered which provides specifications on the structure of a GI-network as well as on the information encoded therein. Let \mathfrak{M}_B denote the model of [BJW⁺10] as depicted in Eq. (2.10)

$$\mathfrak{M}_B : t_{\mathfrak{M}_B} \mapsto T_{\mathfrak{M}_B}. \quad (2.10)$$

where $t_{\mathfrak{M}_B}$ comprises the data types considered and $T_{\mathfrak{M}_B}$ describes the GI-network topology information provided by the model. In particular, the data considered by the model of [BJW⁺10] is comprised of SKs, DKs, and a specific type of knockout measurement based data, denoted as GI-profile data which serves as a measure of the interaction similarity of two specific genes with the rest of the genes under study. Hence, $t_{\mathfrak{M}_B} = (\mathcal{R}_{S_{b,1}}, \mathcal{R}_{S_{b,2}}, t_{\mathfrak{M}_B,3})$ where $t_{\mathfrak{M}_B,3} = \Psi_{\mathfrak{n}}(\mathcal{R})$ denotes the GI-profile data computed by some function $\Psi_{\mathfrak{n}}(\mathcal{R}) \in \mathcal{C}$.

The GI-network topology information $T_{\mathfrak{M}_B}$ is given by the classification of all gene pairs into hierarchical relationship classes which reflect the specific network topology restrictions under the terms of [BJW⁺10], in a different domain. Furthermore, $T_{\mathfrak{M}_B}$ also contains a set of undirected edges that reflects the interaction structure of the undirected skeleton of the underlying GI-network \mathcal{D} which is constrained to follow the structure described in Subsection 2.2.1. In the following, first the data model according to [BJW⁺10] is introduced in detail.

As an important data type in genomics research and systems biology, the GI-profile $\rho(i, j)$ of genes $i, j \in \mathcal{G}$ quantifies the interaction similarity of genes i and j , respectively, with a set of other genes [BJW⁺10]. Note that in the formal context of data models $t_{\mathfrak{M}_B,3} = \Psi_{\mathfrak{n}}(\mathcal{R}) \equiv \{\rho(i, j)\}_{\forall \{i,j\} \in \mathcal{G}}$. There are many measures to quantify the similarity between the interaction profiles of two genes with a set of other genes, however, the most prominent one is the Pearson correlation [Bri95]. As a consequence, the GI-profile data $\rho(i, j)$ is computed as

$$\rho(i, j) = \frac{\sum_{l \in \mathcal{G} \setminus \{i,j\}} (R(i, l) - \bar{R}_i)(R(j, l) - \bar{R}_j)}{\sqrt{\sum_{l \in \mathcal{G} \setminus \{i,j\}} (R(i, l) - \bar{R}_i)^2} \sqrt{\sum_{l \in \mathcal{G} \setminus \{i,j\}} (R(j, l) - \bar{R}_j)^2}} \quad \forall i, j \in \mathcal{G} \quad (2.11)$$

where

$$\bar{R}_i = \frac{1}{G-2} \sum_{l \in \mathcal{G} \setminus \{i,j\}} R(i, l), \quad \bar{R}_j = \frac{1}{G-2} \sum_{l \in \mathcal{G} \setminus \{i,j\}} R(j, l) \quad (2.12)$$

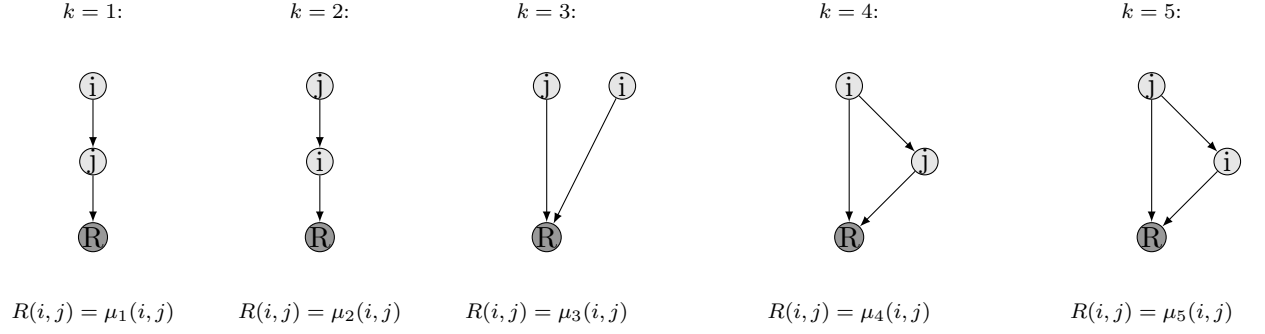


Figure 2.3. Hierarchical relationship classes $k \in \mathcal{K} = \{1, \dots, 5\}$ according to [BJW⁺10] to categorize the genetic interaction type of gene pair i, j along with expected DK phenotype $\mu_k(i, j)$ for each class k .

denote the sample mean of all $G - 2$ DK phenotypes of genes i and j , respectively. Note that the GI-profile data $\rho(i, j)$ as computed according to Eq. (2.11) is in the range of -1 to 1 . Furthermore, it is important to remark that the GI-profile data $\rho(i, j)$ does not need to be computed according to Eq. (2.11). It can also be extracted from some database where a priori knowledge about the set of genes under study, i.e., \mathcal{G} , is stored and manually curated by experts.

In genomics research, it is commonly known that the GI-profile $\rho(i, j)$ is likely to be high if genes i and j interact, i.e., the activity of gene i controls the activity of gene j or the activity of gene j controls the activity of gene i , respectively. From a GI-network/DAG perspective, a high GI-profile $\rho(i, j)$ is a strong indication that genes i and j are linked by an edge. On the contrary, if the GI-profile $\rho(i, j)$ is small then it is not likely that genes i and j interact. Thus from a GI-network/DAG perspective, it is a strong indication that there is no edge linking genes i and j . With respect to their DK phenotypes, the gene pairs $i, j \in \mathcal{G}$ and $j, i \in \mathcal{G}$ are identical, i.e., $R(i, j) = R(j, i)$ and $\rho(i, j) = \rho(j, i)$. Hence, it is sufficient to only consider gene pairs $i, j \in \mathcal{G} : j > i$. However, on behalf of readability and notational convenience the specification that only those gene pairs $i, j \in \mathcal{G}$ are considered, for which $j > i$, is mostly omitted.

According to [BJW⁺10], the genetic interaction of any pair of genes i, j from the set of genes under study, i.e., \mathcal{G} , can be uniquely classified to one out of the five hierarchical relationship classes, which are depicted in Figure 2.3, based on their DK phenotype $R(i, j)$ and their SK phenotypes $R(i)$ and $R(j)$, respectively. The classification of a pair of genes i, j to a specific hierarchical relationship class $k \in \mathcal{K} = \{1, \dots, 5\}$ as depicted in Figure 2.3 models a specific genetic interaction between those two genes which is dictated by the structural/hierarchical relationship between genes i and j in

GI-network/DAG \mathcal{D} that describes the total interactions among the genes in \mathcal{G} . Given that genes $i, j \in \mathcal{G}$ are in class $k \in \mathcal{K}$ in DAG/GI-network \mathcal{D} , that describes the interactions among the investigated set of genes \mathcal{G} , the expected DK phenotype $R(i, j)$ is characterized by the model $\mu_k(i, j)$ which is depicted in Eq. (2.13) by

$$R(i, j) = \begin{cases} \mu_1 = R(j) & \text{if pair } i, j \text{ belongs to class } k = 1 \\ \mu_2 = R(i) & \text{if pair } i, j \text{ belongs to class } k = 2 \\ \mu_3 = R(i)R(j) & \text{if pair } i, j \text{ belongs to class } k = 3 \\ \mu_4 = \frac{1}{2}[R(i)R(j) + R(j)] & \text{if pair } i, j \text{ belongs to class } k = 4 \\ \mu_5 = \frac{1}{2}[R(i)R(j) + R(i)] & \text{if pair } i, j \text{ belongs to class } k = 5 \end{cases} \quad \forall i, j \in \mathcal{G} : j > i \quad (2.13)$$

For each pair of genes $i, j \in \mathcal{G}$, the models $\mu_k(i, j)$, for $k = 1, \dots, 5$ as depicted by Eq. (2.13), relate the DK $R(i, j)$ of gene pair i, j with their respective SKs, i.e., $R(i)$ and $R(j)$. It is important to note that the schematics of the $|\mathcal{K}|$ hierarchical relationship classes of [BJW⁺10] in Figure 2.3 do not reflect absolute adjacency relations between genes i, j in DAG \mathcal{D} , but hierarchical information about the topological relation of those two genes in \mathcal{D} . Hence, given that genes i, j are in class k with respect to DAG \mathcal{D} , it does not follow that genes i and j are arranged in DAG \mathcal{D} according to the schematic of class k . The following three examples elaborate on the hierarchical relationship classes of [BJW⁺10] in Figure 2.3 in order to provide a better understanding about the hierarchical information that is encoded by the classes.

Example 2.2.3. *As indicated by the corresponding schematic in Figure 2.3, the genes $i, j \in \mathcal{G} : j > i$ are in a linear hierarchical relationship with each other in DAG \mathcal{D} in the case of $k = 1$. In particular, the schematic for $k = 1$ illustrates that all paths from gene i to the reporter node R traverse gene j . On the contrary, no paths from gene j to the reporter node R traverse gene i . Hence, the activity of gene j is controlled by the activity of gene i . Due to the fact that gene i only affects the cell function under study via controlling the activity of gene j , the resulting DK phenotype for genes i, j being knocked out, i.e., $R(i, j) = \mu_1(i, j)$, yields the SK $R(j)$ of gene j . It is important to remark that there are in general numerous paths from genes i and j , respectively, to the reporter node R in DAG \mathcal{D} and not only a single one as it could be wrongly concluded by the schematic of class $k = 1$.*

In case of $k = 2$, gene j is in a linear relationship with gene i as well as hierarchically upstream of gene i . With the same line of argument as mentioned above, it follows that the resulting DK phenotype for genes i, j being knocked out, i.e., $R(i, j) = \mu_2(i, j)$, yields the SK $R(i)$ of gene i .

Example 2.2.4. *In the case of $k = 3$, the genes $i, j \in \mathcal{G} : j > i$ in DAG \mathcal{D} are assumed to be independent of each other. As indicated by the corresponding schematic of class*

$k = 3$, there are no paths from gene i to gene j as well as no paths from gene j to gene i in DAG \mathcal{D} . In plain words, there are no paths connecting genes i and j in DAG \mathcal{D} . Hence, the activity of both genes is independent of each other. In this case, the DK phenotype for genes i, j being knocked out is $R(i, j) = \mu_3(i, j) = R(i)R(j)$.

Example 2.2.5. The hierarchical relationship between genes $i, j \in \mathcal{G} : j > i$ in DAG \mathcal{D} depicted by the schematic of class $k = 4$ in Figure 2.3 reflects a combination of classes $k = 1$ and $k = 2$, respectively, with class $k = 3$. Given that genes i, j are in class $k = 4$, then there is at least one path from gene i to gene j in DAG \mathcal{D} . Furthermore, there is at least one path from gene i to the reporter node R that is independent of gene j , i.e., gene j is not traversed by that particular path(s). As a combination of classes $k = 1$ and $k = 3$, the expected DK phenotype for genes i, j in class $k = 4$ is given by $R(i, j) = \mu_4(i, j) = \frac{1}{2}[R(i)R(j) + R(j)]$.

In a similar fashion, the hierarchical relationship class $k = 5$ as depicted in Figure 2.3 models a combination of classes $k = 2$ and $k = 3$, i.e., gene j is hierarchically upstream of gene i in DAG \mathcal{D} . With the same line of argument as mentioned above to justify the DK phenotype of genes i, j being in class $k = 4$, the expected DK phenotype for genes i, j being in class $k = 5$ is given by $R(i, j) = \mu_5(i, j) = \frac{1}{2}[R(i)R(j) + R(i)]$.

For all genes $i \in \mathcal{G}$, let $\mathcal{P}_i = \{P_1^{(i)}, \dots, P_I^{(i)}\}$ denote the set containing all I paths $P_\tau^{(i)}$, $\tau \in \{1, \dots, I\}$ in DAG \mathcal{D} that connects gene i with the reporter node R . In order to identify the hierarchical relationship class between two genes $i, j \in \mathcal{G}$ in DAG \mathcal{D} , the five mutually exclusive decision rules C_1 to C_5 displayed in Table 2.1 provide a formal decision basis. In essence, given that genes $i, j \in \mathcal{G}$ are in class k , then only condition C_k is true. On the contrary, given that condition C_k is true for gene pair $i, j \in \mathcal{G}$ then conditions $C_{k'} \forall k' \in \mathcal{K} \setminus \{k\}$ are false. Furthermore, genes $i, j \in \mathcal{G}$ are in a hierarchical relationship in DAG \mathcal{D} that is reflected by class k .

As stated in Eq. (2.14a) of condition C_1 in Table 2.1, a pair of genes $i, j \in \mathcal{G} : j > i$ in DAG \mathcal{D} is classified to class $k = 1$ if the set of vertices $V(P_\tau^{(i)})$ for all paths $P_\tau^{(i)}$ in \mathcal{P}_i contains gene j . In this case, condition C_1 is true and condition C_2 is false due to the acyclicity of DAG \mathcal{D} . Furthermore, condition C_3 cannot be true since, the constraint that gene j is not traversed by any path $P_\tau^{(i)}$ is a direct contradiction to condition C_1 . The left-hand-side (LHS) of condition C_4 , i.e., the term left of the logical AND operator \wedge directly contradicts condition C_1 as well. Hence, condition C_4 is false. Finally, the right-hand-side (RHS) of Eq. (2.14e) in condition C_5 to be true requires that there exists at least one pair of paths $P_\tau^{(i)}$ and $P_\kappa^{(j)}$ for which $P_\tau^{(i)}$ is a sub-path of $P_\kappa^{(j)}$. However, in the case of condition C_1 being true for gene pair i, j , there are only

C_1 : pair $(i, j) \mapsto$ class $k = 1$

$$\forall P_\tau^{(i)} \in \mathcal{P}_i : j \in V(P_\tau^{(i)}) \quad (2.14a)$$

C_2 : pair $(i, j) \mapsto$ class $k = 2$

$$\forall P_\tau^{(j)} \in \mathcal{P}_j : i \in V(P_\tau^{(j)}) \quad (2.14b)$$

C_3 : pair $(i, j) \mapsto$ class $k = 3$

$$\left(\forall P_\tau^{(i)} \in \mathcal{P}_i : j \notin V(P_\tau^{(i)}) \right) \bigwedge \left(\forall P_\kappa^{(j)} \in \mathcal{P}_j : i \notin V(P_\kappa^{(j)}) \right) \quad (2.14c)$$

C_4 : pair $(i, j) \mapsto$ class $k = 4$

$$\left(\exists P_\tau^{(i)} \in \mathcal{P}_i : j \notin V(P_\tau^{(i)}) \right) \bigwedge \left(\exists P_\tau^{(i)} \in \mathcal{P}_i, P_\kappa^{(j)} \in \mathcal{P}_j : V(P_\kappa^{(j)}) \subset V(P_\tau^{(i)}) \right) \quad (2.14d)$$

C_5 : pair $(i, j) \mapsto$ class $k = 5$

$$\left(\exists P_\kappa^{(j)} \in \mathcal{P}_j : i \notin V(P_\kappa^{(j)}) \right) \bigwedge \left(\exists P_\tau^{(i)} \in \mathcal{P}_i, P_\kappa^{(j)} \in \mathcal{P}_j : V(P_\tau^{(i)}) \subset V(P_\kappa^{(j)}) \right) \quad (2.14e)$$

Table 2.1. Mutually exclusive conditions C_1 to C_5 to identify the hierarchical relationship class k of [BJW⁺10] of a pair of genes $i, j \in \mathcal{G} : j > i$ in DAG \mathcal{D}

pairs of paths $P_\tau^{(i)}$ and $P_\kappa^{(j)}$ for which $P_\tau^{(i)}$ is a super-path of $P_\kappa^{(j)}$. Thus, condition C_5 is false as well. In summary, gene pair $i, j \in \mathcal{G} : j > i$ is characterized by class $k = 1$ if all paths from gene i to the reporter node R traverse gene j .

With the same line of argument, it can be shown that only condition C_2 in Table 2.1 is true if genes $i, j \in \mathcal{G} : j > i$ are in class $k = 2$ in DAG \mathcal{D} .

Condition C_3 as stated in Eq. (2.14c) of Table 2.1 is true, if no path $P_\tau^{(i)}$ from gene i to the reporter node R traverses gene j , i.e., $j \notin V(P_\tau^{(i)})$, $\forall P_\tau^{(i)} \in \mathcal{P}_i$, and furthermore, no path $P_\kappa^{(j)}$ from gene j to the reporter node R traverses gene i , i.e., $i \notin V(P_\kappa^{(j)})$, $\forall P_\kappa^{(j)} \in \mathcal{P}_j$. Given that condition C_3 is true, i.e., genes i and j are independent, it is obvious that conditions C_1 and C_2 are false, since both conditions C_1 and C_2 describe a mutual dependency among genes i and j . Furthermore, given that C_3 is true, there cannot be any pair of paths $P_\tau^{(i)}$, $P_\kappa^{(j)}$ for which $V(P_\kappa^{(j)}) \subset V(P_\tau^{(i)})$ and $V(P_\tau^{(i)}) \subset V(P_\kappa^{(j)})$, respectively. This follows from the fact that paths $P_\tau^{(i)}$ and $P_\kappa^{(j)}$ mutually exclude genes j and i , respectively. Hence, conditions C_4 and C_5 are false as well.

As stated in Eq. (2.14d) of condition C_4 in Table 2.1, a pair of genes $i, j \in \mathcal{G} : j > i$ in DAG \mathcal{D} is classified to class $k = 4$ if there exists at least one path $P_\tau^{(i)}$ to the reporter node R which does not traverse gene j and furthermore, there must exist at least one

pair of paths $P_\tau^{(i)}$, $P_\kappa^{(j)}$ for which the set of nodes $V(P_\kappa^{(j)})$ is a subset of $V(P_\tau^{(i)})$. In other words, condition C_4 is true if for each path $P_\kappa^{(j)} \in \mathcal{P}_j$ there is a super-path $P_\tau^{(i)}$ in \mathcal{P}_i and furthermore, there is at least one path $P_\tau^{(i)}$ in \mathcal{P}_i that does not traverse gene j . Given that condition C_4 is true, it directly follows that condition C_1 is false, since Eq. (2.14a) of condition C_1 demands that all paths $P_\tau^{(i)}$ in \mathcal{P}_i traverse gene j which is a direct contradiction to the LHS of Eq. (2.14d) that requires that there is at least one path $P_\tau^{(i)}$ which is independent of gene j , i.e., $j \notin V(P_\tau^{(i)})$. Furthermore, condition C_2 is false as well, since it requires that all paths $P_\tau^{(j)} \in \mathcal{P}_j$ traverse gene i . According to the RHS of Eq. (2.14b), there is at least one path $P_\kappa^{(j)} \in \mathcal{P}_j$ that is a subpath of a specific $P_\tau^{(i)}$, i.e., $V(P_\kappa^{(j)}) \subset V(P_\tau^{(i)})$. Hence, conditions C_4 and C_2 contradict each other. Condition C_3 requires that all paths $P_\tau^{(i)}$ do not traverse gene j and all paths $P_\kappa^{(j)}$ do not traverse gene i . However, in the case of condition C_4 being true there is at least one pair of paths $P_\tau^{(i)}$, $P_\kappa^{(j)}$ for which path $P_\tau^{(i)}$ is a super-path of $P_\kappa^{(j)}$, i.e., path $P_\tau^{(i)}$ traverses gene j . Hence, given that C_4 is true, condition C_3 is false. The RHS of Eq. (2.14e) in condition C_5 requires that there is at least one pair of paths $P_\tau^{(i)}$, $P_\kappa^{(j)}$ for which $P_\kappa^{(j)}$ is a super-path of $P_\tau^{(i)}$, i.e. $V(P_\tau^{(i)}) \subset V(P_\kappa^{(j)})$. However, this is a direct contradiction to Eq. (2.14d) in condition C_4 , since there it is required the opposite, i.e., there is at least one pair of paths $P_\tau^{(i)}$, $P_\kappa^{(j)}$ for which $P_\tau^{(i)}$ is a super-path of $P_\kappa^{(j)}$, i.e. $V(P_\kappa^{(j)}) \subset V(P_\tau^{(i)})$. However, both cannot be true simultaneously due to the acyclicity and the strict of DAG \mathcal{D} . Consequently, in the case of C_4 being true, condition C_5 is false.

With the same line of argument as used to elucidate condition C_4 , it can be shown that only condition C_5 in Table 2.1 is true if genes $i, j \in \mathcal{G} : j > i$ are in class $k = 5$ in DAG \mathcal{D} .

The classification of each gene pair $i, j \in \mathcal{G} : j > i$ in DAG \mathcal{D} to a specific hierarchical relationship class $k \in \mathcal{K}$ can be interpreted as a transformation of DAG \mathcal{D} into the domain of hierarchical relationship classes of [BJW⁺10]. In particular, let

$$\Omega(\mathcal{D}) = \bigcup_{i,j \in \mathcal{G} : j > i} \omega_{i,j} \quad (2.15)$$

denote the representation of DAG \mathcal{D} in the domain of hierarchical relationship classes of [BJW⁺10] where the class indicator coefficients

$$\omega_{i,j} := k \quad \text{if } C_k \text{ is true} \quad (2.16)$$

denote the classification of gene pair $i, j \in \mathcal{G} : j > i$ of DAG \mathcal{D} to class k under the terms of [BJW⁺10].

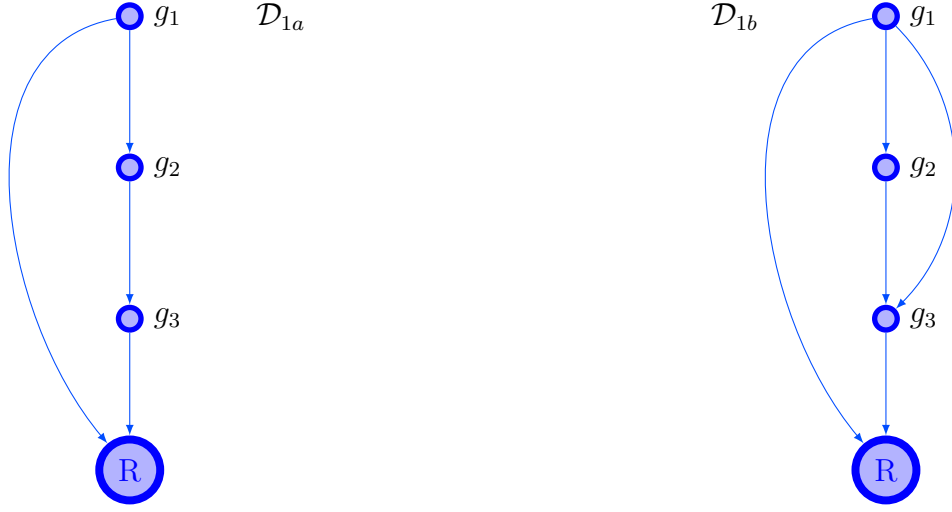


Figure 2.4. Left: DAG \mathcal{D}_{1a} with set of genes $\mathcal{G}_{\mathcal{D}_{1a}} = \{g_1, g_2, g_3\}$ and the set of edges $\mathcal{E}_{\mathcal{D}_{1a}}$ that reflects the displayed topology. Right: DAG \mathcal{D}_{1b} with set of genes $\mathcal{G}_{\mathcal{D}_{1b}} = \{g_1, g_2, g_3\}$ and the set of edges $\mathcal{E}_{\mathcal{D}_{1b}}$ that reflects the displayed topology.

Definition 2.2.1. A DAG-family \mathcal{F}_{Ω_0} denotes the set of all DAGs \mathcal{D} which are represented by Ω_0 in the domain of hierarchical relationship classes. Formally, a DAG-family \mathcal{F}_{Ω_0} is described by

$$\mathcal{F}_{\Omega_0} = \{\mathcal{D} \mid \mathcal{D} \in \mathbb{D}, \Omega(\mathcal{D}) = \Omega_0\} \quad (2.17)$$

where \mathbb{D} denotes the space of all possible DAGs according to the terms of [BJW⁺10].

Example 2.2.6. As displayed in Figure 2.4, the two DAGs \mathcal{D}_{1a} and \mathcal{D}_{1b} have the same set of nodes/genes but a different topology, i.e., $\mathcal{E}_{\mathcal{D}_{1a}} \neq \mathcal{E}_{\mathcal{D}_{1b}}$. Using the conditions in Table 2.1 to identify the hierarchical relationship class k for all gene pairs, the representation of DAG \mathcal{D}_{1a} in the domain of hierarchical relationship classes is given by

$$\Omega(\mathcal{D}_{1a}) = \left\{ \omega_{1,2}^{(1a)}, \omega_{1,3}^{(1a)}, \omega_{2,3}^{(1a)} \right\} \quad (2.18)$$

where $\omega_{1,2}^{(1a)} = 4$, $\omega_{1,3}^{(1a)} = 4$ and $\omega_{2,3}^{(1a)} = 1$. In the same fashion, the representation of DAG \mathcal{D}_{1b} in the domain of hierarchical relationship classes is given by

$$\Omega(\mathcal{D}_{1b}) = \left\{ \omega_{1,2}^{(1b)}, \omega_{1,3}^{(1b)}, \omega_{2,3}^{(1b)} \right\} \quad (2.19)$$

where $\omega_{1,2}^{(1b)} = 4$, $\omega_{1,3}^{(1b)} = 4$ and $\omega_{2,3}^{(1b)} = 1$.

Hence, in the domain of hierarchical relationship classes the two DAGs are equivalent, since $\Omega(\mathcal{D}_{1a}) = \Omega(\mathcal{D}_{1b})$. Thus, DAGs \mathcal{D}_{1a} and \mathcal{D}_{1b} belong to the same DAG family.

In the following, Lemma 2.2.1 is introduced that elaborates on the relationship between a DAG \mathcal{D} in its graph representation according to Eq. (2.7) and its representation in the domain of hierarchical relationship classes $\Omega(\mathcal{D})$ as defined by Eq. (2.15).

Lemma 2.2.1. *Given DAG $\mathcal{D} = (\mathcal{G} \cup \{R\}, \mathcal{E}_{\mathcal{D}})$ with set of genes \mathcal{G} and topology $\mathcal{E}_{\mathcal{D}}$. Let $\Omega_0 = \Omega(\mathcal{D})$ denote the representation of DAG \mathcal{D} in the domain of hierarchical relationship classes. Furthermore, let \mathcal{F}_{Ω_0} denote the DAG-family of \mathcal{D} with $|\mathcal{F}_{\Omega_0}| \geq 1$. Then the reconstruction $\hat{\mathcal{D}}$ of DAG \mathcal{D} based on Ω_0 can be always learned up to DAG-family-equivalence, i.e., $\hat{\mathcal{D}} \in \mathcal{F}_{\Omega_0}$.*

Proof. See Appendix A. □

It is important to remark, that Lemma 2.2.1 implies, that there is in general no unique solution for the reconstruction $\hat{\mathcal{D}}$ of DAG \mathcal{D} based on Ω_0 . As a special case, if and only if $|\mathcal{F}_{\Omega_0}| = 1$, then DAG \mathcal{D} can be uniquely recovered based on Ω_0 .

2.2.3 Class Coupling

Let the SK and DK data $R(i) \forall i \in \mathcal{G}$ and $R(i, j) \forall i, j \in \mathcal{G} : j > i$, respectively, as well as the GI-profile data $\rho(i, j) \forall i, j \in \mathcal{G} : j > i$ be given. Furthermore, assume that the GI-network/DAG \mathcal{D} that reflects the genetic interactions among the set of genes under study, i.e., \mathcal{G} , is unknown. A demanding challenge in estimating the hierarchical relationship class representation $\Omega(\mathcal{D})$ of DAG \mathcal{D} based on the measured SK, DK and GI-profile data is the logical coupling among the class indicator coefficients $\omega_{i,j}$. The strong logical coupling among the class indicator coefficients $\omega_{i,j}$ originates from the fact that the classes $k \in \mathcal{K}$ encode hierarchical topology information about the DAG underlying the observed data. Therefore, gene-pair by gene-pair classification is not applicable without substantial losses in the estimation performance, since this would cause contradicting statements about the topology of DAG \mathcal{D} . In the following examples the problem of the coupled class indicator coefficients $\omega_{i,j}$ is illustrated exemplarily.

Example 2.2.7. *For genes $i, j, l \in \mathcal{G} : l > j > i$, given that $\omega_{i,j} = 1$ and $\omega_{i,l} = 2$ and DAG \mathcal{D} is assumed to be unknown. Then, the gene pair j, l must be in class $k = 2$, i.e., $\omega_{j,l} = 2$, as well. To see this, the topological implications of the classes have to be taken into account. The classification of genes i, j to class $k = 1$ implies that all paths $P_{\tau}^{(i)} \in \mathcal{P}_i$ from gene i to the reporter node R traverse gene j . Thus, there is always a path $P_{\tau}^{(i)}$ that is a superpath of $P_{\kappa}^{(j)}$ for each path $P_{\kappa}^{(j)} \in \mathcal{P}_j$. Furthermore,*

the classification of genes i, l to class $k = 2$ implies that all paths $P_\zeta^{(l)} \in \mathcal{P}_l$ from gene l to the reporter node R traverse gene i . Hence, there is always a path $P_\zeta^{(l)}$ that is a superpath of $P_\tau^{(i)}$ for each $P_\tau^{(i)} \in \mathcal{P}_i$. In consequence, it is always possible to find a path $P_\zeta^{(l)} \in \mathcal{P}_l$ that is a superpath of $P_\kappa^{(j)}$ for each $P_\kappa^{(j)} \in \mathcal{P}_j$. In other words, all paths $P_\zeta^{(l)} \in \mathcal{P}_l$ from gene l to the reporter node R traverse gene j .

Example 2.2.8. For genes $i, j, l \in \mathcal{G} : l > j > i$, assume that $\omega_{i,j} = 2$ and $\omega_{i,l} = 1$ and DAG \mathcal{D} is unknown. Then, the gene pair j, l must be in class $k = 1$, i.e., $\omega_{j,l} = 1$, as well. The classification of genes i, j to class $k = 2$ implies that all paths $P_\kappa^{(j)} \in \mathcal{P}_j$ from gene j to the reporter node R traverse gene i . Thus, there is always a path $P_\kappa^{(j)}$ that is a superpath of $P_\tau^{(i)}$ for each path $P_\tau^{(i)} \in \mathcal{P}_i$. Furthermore, the classification of genes i, l to class $k = 1$ implies that all paths $P_\tau^{(i)} \in \mathcal{P}_i$ from gene i to the reporter node R traverse gene l . Hence, there is always a path $P_\tau^{(i)}$ that is a superpath of $P_\zeta^{(l)}$ for each $P_\zeta^{(l)} \in \mathcal{P}_l$. Consequently, it is always possible to find a path $P_\kappa^{(j)} \in \mathcal{P}_j$ that is a superpath of $P_\zeta^{(l)}$ for each $P_\zeta^{(l)} \in \mathcal{P}_l$. In other words, all paths $P_\kappa^{(j)} \in \mathcal{P}_j$ from gene j to the reporter node R traverse gene l .

It is important to remark that there are many more coupling rules among the class indicator coefficients $\omega_{i,j}$ than stated by the two examples above. In Section 3.1, a formalism to identify all coupling rules is presented and elucidated.

2.3 Annealed Importance Sampling

In order to solve combinatorial problems approximately, metaheuristic algorithms that are based on Markov-Chain-Monte-Carlo (MCMC) sampling are a well established alternative to ILP-formulations. As a state-of-the-art MCMC algorithm to solve combinatorial problems, *simulated annealing* (SA) [KGV83] is a wide-spread algorithm that appears in various fields of applications. For instance, solutions to the famous TSP are often computed by SA-based algorithms [ZLZZ16], [LBL16], [CSMH91]. A prominent adaptation of the SA algorithm in the field of Bioinformatics and genomics research is the *annealed importance sampling* (AIS) algorithm [Nea01] which combines the principles of SA and *importance sampling* (IS). In particular, the AIS algorithm is one of the most important algorithms for learning GI-networks, especially in the context of networks that follow the model of [BJW⁺10]. In the following, first SA is described followed by IS. Finally, the AIS algorithm is presented.

2.3.1 Simulated Annealing

Originally, the idea of SA stems from statistical physics and was developed by [KGV83] in 1983. As a metaheuristic optimization algorithm, inspired by physical processes in the field of metallurgy, SA rapidly found applications in a wide range of application areas, as for instance, in micro-chip layouting [Rut89].

Given the following constrained optimization problem

$$\min_{\mathbf{z}} E(\mathbf{z}) \quad (2.20a)$$

s. t.

$$\mathcal{H} \quad (2.20b)$$

where $E(\cdot)$ denotes an arbitrary objective function that is non-linear and non-convex in general. For the sake of a compact notation, set \mathcal{H} accumulates all constraints which any solution to problem (2.42) has to fulfill. Depending on specifications made in \mathcal{H} , the optimization variable of dimension $n_{\mathbf{z}}$ denoted as \mathbf{z} can belong to different domains, e.g., $\mathbf{z} \in \mathbb{R}^{n_{\mathbf{z}} \times 1}$, $\mathbf{z} \in \mathbb{C}^{n_{\mathbf{z}} \times 1}$, or $\mathbf{z} \in \mathbb{Z}^{n_{\mathbf{z}} \times 1}$. Instead of solving problem (2.42) by numerical optimization methods, as for instance with interior-point methods [KKL⁺07], [HRVW96], [Meh92], in SA problem (2.42) is solved approximately by sampling from a specifically designed Markov-Chain.

Based on the objective function $E(\mathbf{z})$ of problem (2.42), a probability density function (PDF) with respect to \mathbf{z} can be constructed according to Eq. (2.21)

$$p(\mathbf{z}) = \frac{1}{C_N} e^{-E(\mathbf{z})/cT} \quad \mathbf{z} \in \mathcal{H}_{\text{supp}} \quad (2.21)$$

where $C_N = \int_{\mathbf{z} \in \mathcal{H}_{\text{supp}}} e^{-E(\mathbf{z})/cT} d\mathbf{z}$ denotes the normalizing constant of the distribution, $\mathcal{H}_{\text{supp}}$ denotes the support of $p(\mathbf{z})$ which is dictated by the optimization constraints in \mathcal{H} . The constants T and c are non-negative. Due to the physical origin of SA, the objective function $E(\cdot)$ is commonly referred to as “the energy” and the vectors \mathbf{z} are denoted as the “states” of the system. Furthermore, T is often denoted as the “temperature” and c is often chosen as the Boltzmann constant. Note that in an optimization context, constant c is not of interest and can be neglected, i.e., $c = 1$, whereas the temperature T is of high significance. When sampling from the PDF $p(\mathbf{z})$ in Eq. (2.21), the following observations can be made:

- states \mathbf{z} that yield high energy, i.e., $E(\mathbf{z})$ is large for such states, are sampled with lower probability

- states \mathbf{z} that yield low energy, i.e., $E(\mathbf{z})$ is small for such states, are sampled with higher probability
- the minima of $E(\mathbf{z})$ given constraints \mathcal{H} are the modes of $p(\mathbf{z})$

Hence, in order to find the minimum of problem (2.42) the Markov-chain sampling procedure to be constructed must be able to explore the modes of $p(\mathbf{z})$. However, sampling from $p(\mathbf{z})$ is a difficult problem, since computing the normalizing constant C_N is generally intractable due to the following reasons:

- computing C_N implies integrating $p(\mathbf{z})$, which cannot be done in closed form in general
- as an alternative to closed form integration, numerical integration methods can be applied. However, vector \mathbf{z} is generally high dimensional, which makes numerical integration methods become computationally extremely costly and practically intractable for real-life problems. This phenomenon is commonly referred to as *curse of dimensionality*

In essence, the normalizing constant C_N is generally unknown. Furthermore, the PDF $p(\mathbf{z})$ is generally multi-modal with potentially isolated modes. In the following, the construction of a Markov-Chain is described, which overcomes the problems stated above, and is able to sample from $p(\mathbf{z})$ as defined in Eq. (2.21).

The stochastic process that generates a sequence of states $\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(t)}$ is called a Markov-chain if [AdFDJ03]

$$\tilde{p}(\mathbf{z}^{(t)} | \mathbf{z}^{(t-1)}, \dots, \mathbf{z}^{(0)}) = Q(\mathbf{z}^{(t)} | \mathbf{z}^{(t-1)}) \quad (2.22)$$

where $Q(\mathbf{z}^{(t)} | \mathbf{z}^{(t-1)})$ denotes the transition kernel that generates a new state $\mathbf{z}^{(t)}$ given an old state $\mathbf{z}^{(t-1)}$ and $\tilde{p}(\mathbf{z}^{(t)} | \mathbf{z}^{(t-1)}, \dots, \mathbf{z}^{(0)})$ denotes the conditional PDF of Eq. (2.21). The Markov-chain in Eq. (2.22) is said to converge to its *invariant distribution* $p(\mathbf{z}^{(t+1)})$ if

$$\int p_t(\mathbf{z}^{(t)}) Q(\mathbf{z}^{(t+1)} | \mathbf{z}^{(t)}) d\mathbf{z}^{(t)} = p_{t+1}(\mathbf{z}^{(t+1)}) = p_t(\mathbf{z}^{(t+1)}) = p(\mathbf{z}^{(t+1)}) \quad (2.23)$$

holds, i.e., PDF $p_t(\mathbf{z}^{(t+1)}) = p(\mathbf{z}^{(t+1)})$ is an eigenfunction of the integral transform using kernel $Q(\mathbf{z}^{(t+1)} | \mathbf{z}^{(t)})$. Intuitively, the statement in Eq. (2.23) can be understood in the following way. At some iteration t of the Markov Chain, the state $\mathbf{z}^{(t)}$ is sampled from PDF $p_t(\mathbf{z}^{(t)})$. The state $\mathbf{z}^{(t+1)}$ of the next iteration, i.e., $t + 1$, is generated from

kernel $Q(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)})$ given the current state, i.e., $\mathbf{z}^{(t)}$. If irrespectively of the current state, i.e., $\mathbf{z}^{(t)}$, the new state, i.e., $\mathbf{z}^{(t+1)}$, which has been generated by $Q(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)})$, follows the same PDF as the current state, then the Markov-chain has converged to a stationary PDF that is $p(\mathbf{z})$.

In order to find a kernel PDF $Q(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)})$ such that the Markov-chain converges to an invariant distribution that is the desired PDF depicted in Eq. (2.21), the Metropolis-Hastings algorithm [Has70], [MRR⁺53], as the most prominent choice, is utilized. The transition from state $\mathbf{z}^{(t)}$ to the next state $\mathbf{z}^{(t+1)}$ is based on a two-stage procedure. First, a candidate state \mathbf{z}' is sampled from a *proposal* distribution $q_p(\mathbf{z}'|\mathbf{z}^{(t)})$ given the current state, i.e., $\mathbf{z}^{(t)}$. In the second step, the generated candidate state \mathbf{z}' is accepted as the new state, i.e., as the state of iteration $t + 1$, with probability $A_c(\mathbf{z}^{(t)}, \mathbf{z}')$. According to Metropolis-Hastings, the acceptance probability $A_c(\mathbf{z}^{(t)}, \mathbf{z}')$ is given by

$$A_c(\mathbf{z}^{(t)}, \mathbf{z}') = \min \left\{ 1, \frac{p(\mathbf{z}')q_p(\mathbf{z}^{(t)}|\mathbf{z}')}{p(\mathbf{z}^{(t)})q_p(\mathbf{z}'|\mathbf{z}^{(t)})} \right\} \quad (2.24)$$

in order to guarantee that invariant distribution of the constructed Markov-chain is the desired PDF of Eq. (2.21). The resulting PDF kernel $Q(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)})$ is given by

$$Q(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)}) = q_p(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)})A_c(\mathbf{z}^{(t)}, \mathbf{z}^{(t+1)}) + \delta_{\mathbf{z}^{(t)}}(\mathbf{z}^{(t+1)})r(\mathbf{z}^{(t)}) \quad (2.25)$$

where $\delta_{\mathbf{z}^{(t)}}(\mathbf{z}^{(t+1)})$ is 1 if $\mathbf{z}^{(t+1)} \equiv \mathbf{z}^{(t)}$ and 0 otherwise. The residual $r(\mathbf{z}^{(t)})$ denotes the probability of staying in state $\mathbf{z}^{(t)}$ at iteration $t + 1$, due to the case that the proposed candidate state $\mathbf{z}^{(t)}$ for iteration $t + 1$ can be refused, and is given by

$$r(\mathbf{z}^{(t)}) = \int_{\mathcal{H}_{\text{supp}}} q_p(\mathbf{z}'|\mathbf{z}^{(t)})(1 - A_c(\mathbf{z}^{(t)}, \mathbf{z}'))d\mathbf{z}'. \quad (2.26)$$

Straightforwardly using the Metropolis-Hastings algorithm to design a transition kernel $Q(\mathbf{z}^{(t+1)}|\mathbf{z}^{(t)})$ such that the enrolled Markov-chain converges to $p(\mathbf{z})$ of Eq. (2.21), bears some problems. In theory, the Metropolis-Hastings algorithm is able to sample from multi-modal distributions, however, only in the asymptotic regime where the number of Markov-chain iteration tends to infinity. In practice, the Metropolis-Hastings algorithm is caught in the closest mode without any chance of leaving this particular mode. This is an undesired behaviour in an optimization context, since the finding the solution to an optimization problem implies exploring all modes.

In order to overcome this problem, the Metropolis-Hastings algorithm is modified by a *cooling schedule* of temperatures. In essence, a sequence of temperatures $T^{(0)}, \dots, T^{(t)}$ associated with the corresponding Markov-chain iterations is introduced, where $T^{(0)} \in$

\mathbb{R}_+ and $\lim_{t \rightarrow \infty} T^{(t)} = 0$. At iteration t of the Markov-chain, the target distribution $p(\mathbf{z})$ of Eq. (2.21) is modified according to Eq. (2.27)

$$p(\mathbf{z}) = \frac{1}{C_N} e^{-E(\mathbf{z})/T^{(t)}} \quad \mathbf{z} \in \mathcal{H}_{\text{supp}} \quad (2.27)$$

with $c = 1$. Note that in practice, the Markov-chain is run a predefined number of times denoted as N_{MC} . Furthermore, $T^{(0)}$ is usually larger than 1 and N_{MC} approaches 0.

The idea behind the cooling procedure from high temperatures, i.e., $T^{(0)}$, to low temperatures, i.e., $T^{(N_{\text{MC}})}$, is the following. In the high temperature regime, the target PDF $p(\mathbf{z})$ in Eq. 2.27 is very smooth and, if any, only very weakly multi-modal. Hence, in the high temperature stages, the simulated Markov-chain can easily explore the entire state space, i.e., $\mathcal{H}_{\text{supp}}$, being able to explore all modes. This stage of the Markov-chain is often referred to as the “the exploration phase”.

With increasing temperature, i.e., with increasing number of iterations, the PDF in Eq. 2.27 is scaled by the temperatures in such a way that the probability mass increasingly concentrates around the modes, i.e., the modes of the PDF become more and more isolated. This in turn has the effect, that in the lower/low temperature scenario, the Markov chain gets stuck in some mode of the target PDF not being able to leave it. In this case, the Markov-chain explores only the mode in which it is located in. Thus, highly likely states \mathbf{z} are sampled that are in turn feasible points to problem (2.42) exhibiting small values of the objective function $E(\mathbf{z})$ in problem (2.42). This stage of the Markov-chain is commonly termed as the “exploitation phase”.

The pseudo-code of the SA algorithm is depicted in Algorithm 1

Algorithm 1 Simulated Annealing Algorithm

- 1: **Initialize:**
 $\mathbf{z}^{(0)}, T^{(0)}$
 - 2: **for** $t = 1, \dots, N_{\text{MC}} - 1$ **do**
 - 3: Sample u from uniform distribution, i.e., $u \sim \mathcal{U}(0, 1)$
 - 4: Sample \mathbf{z}' from $q_{\text{p}}(\mathbf{z}' | \mathbf{z}^{(t)})$
 - 5: **if** $u < A_{\text{c}}(\mathbf{z}^{(t)}, \mathbf{z}') = \min \left\{ 1, \frac{p^{1/T^{(t)}}(\mathbf{z}') q_{\text{p}}(\mathbf{z}^{(t)} | \mathbf{z}')}{p^{1/T^{(t)}}(\mathbf{z}^{(t)}) q_{\text{p}}(\mathbf{z}' | \mathbf{z}^{(t)})} \right\}$ **then**
 - 6: Set $\mathbf{z}^{(t+1)} = \mathbf{z}'$
 - 7: **else**
 - 8: Set $\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)}$
 - 9: Set $T^{(t+1)}$ according to predefined cooling schedule
-

An approximate solution $\mathbf{z}_{\text{SA, opt}}$ to problem (2.42) can be found according to Eq. (2.28)

$$\mathbf{z}_{\text{SA, opt}} = \arg \max_{\mathbf{z}^{(t)}: t=0, \dots, N_{\text{MC}}-1} p(\mathbf{z}^{(t)}). \quad (2.28)$$

It is important to remark, that the cooling schedule is a user-defined choice and affects the performance of the SA algorithm significantly. Furthermore, the SA algorithm is generally very slow due to a potentially large number of Markov-chain iterations N_{MC} which are necessary in order to approximately solve problems with many variables. Although being a very powerful method for optimization, the SA algorithm is not suitable to generate i.i.d. samples of the desired target distribution for the purpose of statistical inference. In the high temperature regime, the samples generated according to Algorithm 1 are drawn from smoothed versions of the desired target distribution $p(\mathbf{z})$ whereas in the low temperature regime, the samples are drawn from degenerate distributions $p^{1/T^{(t)}}(\mathbf{z}')$ for which the probability mass is heavily centered in the modes. In essence, the samples generated according to Algorithm 1 yield considerable results in an optimization context, however, the generated sequence of points $\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(N_{\text{MC}}-1)}$ does not follow the desired target distribution $p(\mathbf{z})$.

2.3.2 Importance Sampling

In statistics, a frequently encountered problem is the computation of integrals of the form

$$I(f_{\text{is}}) = \int f_{\text{is}}(\mathbf{z}) \check{p}(\mathbf{z}) d\mathbf{z} \quad (2.29)$$

where $f_{\text{is}}(\cdot)$ denotes some function of interest and $\check{p}(\mathbf{z})$ denotes a PDF. Note that the integral in Eq. (2.29) can also be written as the expected value of $f_{\text{is}}(\mathbf{z})$ with respect to PDF $\check{p}(\mathbf{z})$, i.e., $\mathbb{E}_{\check{p}(\mathbf{z})}[f_{\text{is}}(\mathbf{z})]$. For most practical scenarios, the direct computation of $I(f_{\text{is}})$ is computationally intractable, due to the large state space of \mathbf{z} . This problem is commonly referred to as “the curse of dimensionality” [CL01]. Instead of directly computing $\mathbb{E}_{\check{p}(\mathbf{z})}[f_{\text{is}}(\mathbf{z})]$ in (2.29), a better approach is to sample a sequence $\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(N_{\text{IS}})}$ of N_{IS} points from $\check{p}(\mathbf{z})$ and estimate $\mathbb{E}_{\check{p}(\mathbf{z})}[f_{\text{is}}(\mathbf{z})]$ based on the sampled sequence of points. However, sampling from $\check{p}(\mathbf{z})$ can be computationally very costly and difficult, for instance in the case of $\check{p}(\mathbf{z})$ being multi-modal and high-dimensional.

As a well known method for Monte-Carlo integration [Gew89], importance sampling uses a proposal distribution $\check{q}(\mathbf{z})$ to approximate $I(f_{\text{is}})$ by sampling from this proposal distribution instead of sampling from the original distribution $\check{p}(\mathbf{z})$ [GI89], [RK16]. The proposal distribution must fulfill the following two properties:

- the support of $\check{q}(\mathbf{z})$ includes the support of $\check{p}(\mathbf{z})$
- sampling from $\check{q}(\mathbf{z})$ is easy

Given a suitable proposal distribution $\check{q}(\mathbf{z})$, $I(f_{\text{is}})$ can be written as

$$I(f_{\text{is}}) = \int f_{\text{is}}(\mathbf{z})w(\mathbf{z})\check{q}(\mathbf{z})d\mathbf{z} \quad (2.30)$$

where $w(\mathbf{z}) = \frac{\check{p}(\mathbf{z})}{\check{q}(\mathbf{z})}$ is denoted as the importance weight [AdFDJ03]. Given a sequence of points $\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(N_{\text{IS}})}$ generated from $\check{q}(\mathbf{z})$, $I(f_{\text{is}})$ can be approximated as

$$\hat{I}(f_{\text{is}}) = \sum_{t=0}^{N_{\text{IS}}} f_{\text{is}}(\mathbf{z}^{(t)})w(\mathbf{z}^{(t)}). \quad (2.31)$$

Furthermore, the approximation $\hat{I}(f_{\text{is}})$ can be also interpreted as $I(f_{\text{is}})$ integrated with respect to distribution

$$\check{p}_{N_{\text{IS}}}(\mathbf{z}) = \sum_{t=0}^{N_{\text{IS}}} w(\mathbf{z}^{(t)})\delta_{\mathbf{z}^{(t)}}(\mathbf{z}) \quad (2.32)$$

instead of $\check{p}(\mathbf{z})$, where

$$\delta_{\mathbf{z}^{(t)}}(\mathbf{z}) = \begin{cases} \infty & \text{if } \mathbf{z} = \mathbf{z}^{(t)} \\ 0 & \text{otherwise} \end{cases}. \quad (2.33)$$

Hence, the PDF in $\check{p}_{N_{\text{IS}}}(\mathbf{z})$ in Eq. (2.32) is an approximation of the true PDF $\check{p}(\mathbf{z})$ based on the sampled sequence of points $\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(N_{\text{IS}})}$ [AdFDJ03]. Note that up to now, it has always been assumed that $\check{p}(\mathbf{z})$ is known completely. However, in many practical scenarios PDF $\check{p}(\mathbf{z})$ is only known up to proportionality, i.e., the normalizing constant is unknown. In this case, the approximation of $I(f_{\text{is}})$ is given by

$$\tilde{I}(f_{\text{is}}) = \frac{\sum_{t=0}^{N_{\text{IS}}} f_{\text{is}}(\mathbf{z}^{(t)})w(\mathbf{z}^{(t)})}{\sum_{t=0}^{N_{\text{IS}}} w(\mathbf{z}^{(t)})} = \sum_{t=0}^{N_{\text{IS}}} f_{\text{is}}(\mathbf{z}^{(t)})\tilde{w}(\mathbf{z}^{(t)}) \quad (2.34)$$

with normalized importance weights $\tilde{w}(\mathbf{z}^{(t)})$. As presented above, the idea of Monte-Carlo integration and empirical density estimation is based on a sequence of samples. The more this generated sequence of samples follows the distribution of interest $\check{p}(\mathbf{z})$, the better the approximations will be. Since the samples are not drawn from $\check{p}(\mathbf{z})$ directly, the functionality of the importance weights is to compensate to some extent for the fact that the frequency of some samples in the generated sequence contradicts $\check{p}(\mathbf{z})$. IS is a useful and widespread tool in statistics for Monte-Carlo integration and empirical density estimation. The performance of the importance sampling method critically depends on an appropriate choice of $\check{q}(\mathbf{z})$. However, in many practical scenarios, finding a suitable proposal distribution $\check{q}(\mathbf{z})$ is very difficult and sometimes it is proverbially like finding the needle in the haystack.

2.3.3 Annealed Importance Sampling

Annealed importance sampling [Nea01] combines the ideas of simulated annealing and importance sampling in order to construct a Markov-chain sampling algorithm that takes account of the following two aspects:

- the constructed Markov-chain sampler is able to sample from highly multi-modal probabilities and probability densities with potentially isolated modes. Hence, this Markov-chain sampler is suitable for finding approximate solutions to optimization problems of the form of (2.42) that can be related to distributions according to in Eq. (2.21)
- the generated sequence of samples is suitable for statistical inference, i.e., all samples follow the desired target distribution

The AIS algorithm of [Nea01] constructs a non-homogeneous Markov-chain with $N_{\text{MC}} + 1$ tempered transitions, similar to Algorithm 1, in order to define a suitable proposal distribution $\check{q}(\mathbf{z}_{N_{\text{MC}}}, \dots, \mathbf{z}_0)$ for an importance sampler. Ultimately, only the last generated sample of the Markov-chain is of interest for the importance sampling procedure. In particular, the SA-based Markov-chain is used to generate a sequence of $N_{\text{AR}} + 1$ i.i.d. importance samples denoted as $\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(t')}, \dots, \mathbf{z}^{N_{\text{AR}}}$ where the t' -th importance sample, i.e., $\mathbf{z}^{(t')}$, is equivalent to the last sample of the constructed Markov chain, i.e., $\mathbf{z}^{(t')} \equiv \mathbf{z}_{N_{\text{MC}}}$. In essence, AIS is a nested combination of IS and SA where a SA-based Markov-chain is constructed in order to simulate a suitable proposal distribution for the IS algorithm.

Note that from now on, the normalizing constant of the target distribution is assumed to be unknown meaning that the target distribution is only known up to proportionality. In the following, the construction of the SA-based Markov-chain is described.

Let $p(\mathbf{z}) \propto f_{N_{\text{MC}}}(\mathbf{z})$ denote the multi-modal target distribution of, which the IS proposal distribution $\check{q}(\mathbf{z}_{N_{\text{MC}}}, \dots, \mathbf{z}_0)$ is designed to draw samples from, where $p(\mathbf{z})$ is only known up to proportionality, i.e., up to $f_{N_{\text{MC}}}(\mathbf{z})$. Furthermore, let $p_0(\mathbf{z}) \propto f_0(\mathbf{z})$ denote a simple “initial” distribution of the SA-based Markov-chain that is easy to sample from and has the same support as $p(\mathbf{z})$.

The stationary distribution at iteration t of the simulated SA-based Markov-chain is given by

$$p_t(\mathbf{z}) \propto f_t(\mathbf{z}) = f_{N_{\text{MC}}}(\mathbf{z})^{T^{(t)}} f_0(\mathbf{z})^{1-T^{(t)}} \quad t = 0, \dots, N_{\text{MC}} \quad (2.35)$$

where $T^{(t)}$ denotes the temperature of the SA-based Markov-chain at iteration t . Furthermore, the sequence of temperatures $T^{(0)}, \dots, T^{(t)}, \dots, T^{(N_{MC})}$ must obey the following cooling schedule property:

$$1 = T^{(N_{MC})} > \dots > T^{(t)} > \dots > T^{(0)} = 0. \quad (2.36)$$

Note that $p_t(\mathbf{z})$ is only known up to proportionality as well. Due to the SA-based construction, the proposal distribution $\check{q}(\mathbf{z}_{N_{MC}}, \dots, \mathbf{z}_0)$ of the IS algorithm is given by

$$\check{q}(\mathbf{z}_{N_{MC}}, \dots, \mathbf{z}_0) = \left(\prod_{t=0}^{N_{MC}} Q_t(\mathbf{z}_{t+1}|\mathbf{z}_t) \right) p_0(\mathbf{z}_0) \quad (2.37)$$

where the transition kernel of the SA-based Markov-chain $Q_t(\mathbf{z}_{t+1}|\mathbf{z}_t)$ that models the transition of moving from state \mathbf{z}_t to state \mathbf{z}_{t+1} is given according Eq. (2.38)

$$Q_t(\mathbf{z}_{t+1}|\mathbf{z}_t) = q_p(\mathbf{z}_{t+1}|\mathbf{z}_t) A_{t,c}(\mathbf{z}_t, \mathbf{z}_{t+1}) + \delta_{\mathbf{z}_t}(\mathbf{z}_{t+1}) r(\mathbf{z}_t). \quad (2.38)$$

Note that transition kernel $Q_t(\mathbf{z}_{t+1}|\mathbf{z}_t)$ in Eq. (2.38) is a traditional Metropolis-Hastings kernel construction, [Has70], [MRR⁺53] where the residual $r(\mathbf{z}_t)$ according to Eq. (2.26) reflects the probability / probability density of staying in state \mathbf{z}_t .

Consequently, the acceptance probability at iteration t is given by

$$A_{t,c}(\mathbf{z}_t, \mathbf{z}_{t+1}) = \min \left\{ 1, \frac{p_t(\mathbf{z}_{t+1}) q_p(\mathbf{z}_t|\mathbf{z}_{t+1})}{p_t(\mathbf{z}_t) q_p(\mathbf{z}_{t+1}|\mathbf{z}_t)} \right\} \quad (2.39)$$

where $q_p(\mathbf{z}_{t+1}|\mathbf{z}_t)$ denotes the proposal distribution of the SA-based Markov-chain.

Given a sequence of N_{AR} points $\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(t')}, \dots, \mathbf{z}^{(N_{AR})}$ that have been generated according to the SA-based Markov-chain elucidated above, i.e., $\mathbf{z}^{(t')} \equiv \mathbf{z}_{N_{MC}}$, $(\mathbf{z}_{N_{MC}}, \dots, \mathbf{z}_{N_0}) \sim \check{q}(\mathbf{z}_{N_{MC}}, \dots, \mathbf{z}_0)$, then the empirical distribution $\check{p}_{N_{AIS}}(\mathbf{z})$ of the overall target distribution $p(\mathbf{z})$ is given by

$$\check{p}_{N_{AIS}}(\mathbf{z}) = \frac{1}{\sum_{t'=0}^{N_{AR}} w(\mathbf{z}^{(t')})} \sum_{t'=0}^{N_{AR}} w(\mathbf{z}^{(t')}) \delta_{\mathbf{z}^{(t')}}(\mathbf{z}) \quad (2.40)$$

with importance weights $w(\mathbf{z}^{(t')})$ given by [Nea01]

$$w(\mathbf{z}^{(t')}) = \frac{f_1(\mathbf{z}_0) f_2(\mathbf{z}_1) \dots f_{N_{MC}-1}(\mathbf{z}_{N_{MC}-1}) f_{N_{MC}}(\mathbf{z}_{N_{MC}})}{f_0(\mathbf{z}_0) f_1(\mathbf{z}_1) \dots f_{N_{MC}-2}(\mathbf{z}_{N_{MC}-1}) f_{N_{MC}-1}(\mathbf{z}_{N_{MC}})}. \quad (2.41)$$

In Algorithm 2, the AIS procedure is summarized.

The authors of [BJW⁺10] utilized the illustrated AIS algorithm of [Nea01] in order to generate a sequence of GI-networks $\mathcal{D}^{(0)}, \dots, \mathcal{D}^{(N_{AR})}$ that are of the special topology

of [BJW⁺10] which is described in Subsection 2.2.1. Hence, the states \mathbf{z}_t of the SA based Markov-chain, as well as the generated importance samples $\mathbf{z}^{t'}$ contain both: the topology information of the sampled GI-networks and their respective hierarchical relationship class representation.

Due to the special problem structure, the authors of [BJW⁺10] propose the following distributions:

- initial “simple” distribution of the SA based Markov-chain, i.e., $p_0(\mathbf{z})$, is given by the uniform distribution over all possible GI-networks that follow the DAG topology rules described in Subsection 2.2.1
- given a GI-network represented by \mathbf{z}_t , the proposal distribution $q_p(\mathbf{z}'|\mathbf{z}_t)$ proposes a modification to GI-network/state \mathbf{z}_t by applying, with equal probability, one of the following structure modifications:
 - inserting a node/gene into a linear pathway
 - popping a node out of a linear pathway
 - swapping the order of two nodes in a linear pathway
 - detaching an entire linear pathway and reattaching it elsewhere in the network
 - adding an edge
 - deleting an edge

Thus, in each SA-based Markov-chain iteration t , the proposal distribution $q_p(\mathbf{z}'|\mathbf{z}_t)$ is the uniform distribution over legal networks that can be generated by applying one of the above mentioned structure modifications to the current state/GI-network \mathbf{z}_t .

Algorithm 2 Annealed Importance Sampling

```
1: Initialize:  
    $N_{\text{MC}}, N_{\text{AR}}$ , cooling schedule  $\{T^{(0)}, \dots, T^{(N_{\text{MC}})}\}$   
2: for  $t' = 0, \dots, N_{\text{AR}}$  do  
3:   for  $t = 0, \dots, N_{\text{MC}}$  do  
4:     Sample  $u$  from uniform distribution, i.e.,  $u \sim \mathcal{U}(0, 1)$   
5:     Sample  $\mathbf{z}'$  from  $q_{\text{p}}(\mathbf{z}' | \mathbf{z}^{(t)})$   
6:     if  $u < A_{\text{c}}(\mathbf{z}_t, \mathbf{z}') = \min \left\{ 1, \frac{p_t(\mathbf{z}') q_{\text{p}}(\mathbf{z}_t | \mathbf{z}')}{p_t(\mathbf{z}_t) q_{\text{p}}(\mathbf{z}' | \mathbf{z}_t)} \right\}$  then  
7:       Set  $\mathbf{z}_{t+1} = \mathbf{z}'$   
8:     else  
9:       Set  $\mathbf{z}_{t+1} = \mathbf{z}_t$   
10:    Set  $T^{(t+1)}$  according to predefined cooling schedule  
11:    set  $\mathbf{z}^{(t')} \equiv \mathbf{z}_{N_{\text{MC}}}$ ,  $(\mathbf{z}_{N_{\text{MC}}}, \dots, \mathbf{z}_{N_0}) \sim \check{q}(\mathbf{z}_{N_{\text{MC}}}, \dots, \mathbf{z}_0)$   
12:    compute  $w(\mathbf{z}^{(t')})$  according to (2.41)  
return  $\{\mathbf{z}^{(0)}, \dots, \mathbf{z}^{(N_{\text{AR}})}\}$ 
```

2.4 Integer Programming

The optimization problems dealt with in the field of integer programming are generally of a combinatorial nature. In particular, given an optimization objective subject to a set of constraints with respect to some optimization variable, the goal is to find an integer solution of that specific optimization variable. Given that an integer program, i.e., an optimization problem in discrete variables subject to a set of constraints, is purely linear in its optimization variables with respect to the optimization objective and the constraints, then it is called an *integer linear program* (ILP) [CBD11], [PW06]. Probably the most prominent ILP is the famous TSP where a salesman is to find the shortest route in order to visit a predefined collection of cities with the restriction that each city is visited at most once. As a fundamental representative of the class of ILPs, the TSP has numerous applications from a wide range of branches, as for instance, logistics, production planning and DNA sequencing. Due to their combinatorial nature, ILPs are generally classified as non-deterministic polynomial time (NP)-hard which means that they cannot be solved in polynomial time. Moreover, ILPs are known to scale very poorly with the problem size. In order to solve an ILP, there have been developed several methods. The most prominent ones are the branch-and-bound (BB) and branch-and-cut (BC) methods, [CBD11], [PW06], [LW66], [PR91], [AKM05], where the BC method combines the ideas of the BB algorithm with some principles of the cutting-plane-approach and the group-theoretic-approach [MMWW02], [RD10]. The BC algorithm can be seen as a generalization of the BB method that strongly relies on the ideas of the BB method. Furthermore, BB methods have a significant importance for a variety of practical applications, as for instance, in the field of mobile communication networks of the fourth and fifth generation. There, the design of electromagnetic wavefronts radiated from the antennas is based on integer and mixed integer problems, respectively. This process is commonly referred to as beamforming and intensively studied in [CPP13], [CP13], [CP15], [CDPP12a], [CDPP12b], [CPP12]. Due to its importance for solving integer and mixed integer problems, the basic concept of the BB method is briefly summarized in the following. Note that the explanation of the BB method in this section is mainly based on the book of [CBD11].

In general, the fundamental idea behind the BB method is to design a process that divides the original problem into a sequence of subproblems that are easier to solve. This strategy is commonly referred to as *divide-and-conquer*. Given integer optimization problem \mathcal{O} of the following form

$$\max_{\mathbf{z}} E(\mathbf{z}) \quad (2.42a)$$

s. t.

$$\begin{aligned} & \mathcal{H} \\ & \mathbf{z} \in \mathbb{Z}^{n_z \times 1} \end{aligned} \quad (2.42b)$$

where $E(\cdot)$ denotes an objective function that convex. For the sake of a compact notation, set \mathcal{H} accumulates all constraints which any solution to problem (2.42) has to fulfill. Furthermore, assume that the constraints in \mathcal{H} span a convex set. Note that the general maximization problem \mathcal{O} can be easily converted into a minimization problem by multiplying the objective function $E(\mathbf{z})$ with -1 .

A convenient way of representing the BB algorithm, which solves problem \mathcal{O} , is via a specialized enumeration tree that keeps track of the cascade of splitted subproblems, of their solutions and of their associated upper/lower bounds, which are used as decision metrics. In essence, the BB algorithm performs an implicit enumeration of all possible solutions of problem \mathcal{O} based on an enumeration tree that efficiently discards suboptimal solutions while being constructed in an interactive fashion. The BB procedure involves global and local properties. The global properties are:

- tuple $(\mathbf{z}^*, E(\mathbf{z}^*))$ that denotes the solution to \mathcal{O}_{cr} which is the continuous relaxation of the original integer program (IP) \mathcal{O}
- \bar{b} lowest (best) upper bound on $E(\mathbf{z}^*)$ of the given IP \mathcal{O} , i.e., initially $\bar{b} \equiv E(\mathbf{z}^*)$
- \underline{b} highest (best) lower bound on $E(\mathbf{z}^*)$ of the given IP \mathcal{O} , i.e., $\underline{b} \equiv -\infty$ as long as no integer solution has been found for \mathcal{O}
- The tree is of a “top-down” topology, i.e., the highest/top node in hierarchy of the tree is its root, while the leafs of the tree are the lowest nodes in hierarchy.

Note that the highest (best) lower bound, i.e., \underline{b} , is constantly updated during the evaluation of the BB enumeration tree. Furthermore, the structure of the branch-and-bound tree is characterized by the following local properties:

- Each node \tilde{k} in the tree represents a subproblem $\mathcal{O}^{\tilde{k}}$ of \mathcal{O} and is associated with the following properties: a continuous relaxation to $\mathcal{O}^{\tilde{k}}$ denoted as $\mathcal{O}_{\text{cr}}^{\tilde{k}}$, a solution to problem $\mathcal{O}_{\text{cr}}^{\tilde{k}}$ denoted as tuple $(\mathbf{z}^{\tilde{k}}, E(\mathbf{z}^{\tilde{k}}))$, the lowest (best) upper bound on $\mathcal{O}^{\tilde{k}}$ denoted as $\bar{b}^{\tilde{k}} \equiv E(\mathbf{z}^{\tilde{k}})$, the highest (best) lower bound $\underline{b}^{\tilde{k}}$ on problem \mathcal{O} found so far

- The edges are commonly referred to as “branches”. The branches correspond to specific constraints that separate/split (cut) the subproblems from their parent subproblems. From a solution space perspective, each branching/split partitions the solution space of its parent subproblem, i.e., $\mathcal{O}^{\tilde{k}}$, into two disjoint solution spaces, each associated with one of the created subproblems, i.e., $\mathcal{O}^{\tilde{k}+1}$ and $\mathcal{O}^{\tilde{k}+2}$, respectively. The union of the so generated solution spaces contains the same integer points as the solution space of the parent subproblem, but not the entire solution space of their parent subproblem $\mathcal{O}^{\tilde{k}}$. Hence, regions that cannot contain integer solutions are cut off.
- The root node represents the original problem, i.e., problem $\mathcal{O} \equiv \mathcal{O}^{\tilde{k}}$ for $\tilde{k} = 0$.
- The leafs either represent candidate solutions to problem \mathcal{O} or terminal nodes in the sense that (a) the associated subproblem is infeasible or (b) the lowest upper bound on $\mathcal{O}^{\tilde{k}}$, i.e., $\bar{b}^{\tilde{k}}$, is lower as \underline{b} which is the best lower bound on \mathcal{O} found so far.

Using the above presented properties and notation, the BB algorithm is briefly described in the following steps according to [CBD11].

step 0 Initialization:

Solve the continuous relaxation of IP \mathcal{O} , i.e., \mathcal{O}_{cr} distinguishing the following cases:

- (a) \mathcal{O}_{cr} is infeasible, IP \mathcal{O} is infeasible as well \rightarrow terminate the algorithm and give a certificate of infeasibility.
- (b) The solution to \mathcal{O}_{cr} is integer. Then, problem \mathcal{O} is solved and the algorithm terminates.
- (c) Otherwise, the best upper bound \bar{b} is initialized as the objective value of the solution to \mathcal{O}_{cr} , where the best lower bound \underline{b} is set to $\underline{b} = -\infty$.

step 1 Node-Selection:

Select a node \tilde{k} and its associated subproblem, i.e., $\mathcal{O}^{\tilde{k}}$, of the current BB enumeration tree according to a predefined tree exploration strategy. Use, for instance, depth-first, best-bound-first [CBD11].

step 2 Updating Upper Bound:

Solve $\mathcal{O}_{\text{cr}}^{\tilde{k}} \rightarrow$ obtain $(\mathbf{z}^{\tilde{k}}, E(\mathbf{z}^{\tilde{k}}))$. Set the lowest upper bound $\bar{b}^{\tilde{k}}$ on $\mathcal{O}^{\tilde{k}}$ to $E(\mathbf{z}^{\tilde{k}})$

step 3 Pruning by Infeasibility:

If $\mathcal{O}_{\text{cr}}^{\tilde{k}}$ is infeasible, prune the current node and go to step 1. Otherwise, move to step 4.

step 4 **Pruning by Bound:**

Assuming that $E(\mathbf{z}^{\tilde{k}}) \leq \underline{b}$ holds. Then, exploring the BB tree from node \tilde{k} downwards cannot yield a better solution than the current highest (best) lower bound on the original problem, i.e., \mathcal{O} . Hence, prune the current node and move to step 1. Otherwise, move to step 5.

step 5 **Pruning by Optimality:**

- (a) If $\mathbf{z}^{\tilde{k}}$ is integer, $\mathbf{z}^{\tilde{k}}$ is a candidate solution to problem \mathcal{O} . Set highest lower bound $\underline{b}^{\tilde{k}} = E(\mathbf{z}^{\tilde{k}})$ and compare $\underline{b}^{\tilde{k}}$ with \underline{b} . Consider two cases: (i) if $\underline{b}^{\tilde{k}} \geq \underline{b}$, then set $\underline{b} = \underline{b}^{\tilde{k}}$ (ii) otherwise, \underline{b} remains unchanged. Furthermore, the node \tilde{k} is pruned in the BB enumeration tree, since no better solution on subproblem $\mathcal{O}^{\tilde{k}}$ can be found by splitting $\mathcal{O}^{\tilde{k}}$ into further subproblems. Move to step 1.
- (b) If $\mathbf{z}^{\tilde{k}}$ is non-integer, move to step 6.

step 6 **Branching:**

Select a variable $z_k^{\tilde{k}}$ with fractional value from $\mathbf{z}^{\tilde{k}} = (z_0^{\tilde{k}}, \dots, z_{n_z}^{\tilde{k}})$ of the current node \tilde{k} . Then split the solution space of $\mathcal{O}^{\tilde{k}}$ into two subproblems $\mathcal{O}^{\tilde{k}+1}$ and $\mathcal{O}^{\tilde{k}+2}$ according to the following cuts:

$$\mathcal{O}^{\tilde{k}+1} = \mathcal{O}^{\tilde{k}} \cap \{\mathbf{z} : z_k \leq \lfloor z_k^{\tilde{k}} \rfloor\} \quad (2.43)$$

$$\mathcal{O}^{\tilde{k}+2} = \mathcal{O}^{\tilde{k}} \cap \{\mathbf{z} : z_k \geq \lceil z_k^{\tilde{k}} \rceil\} \quad (2.44)$$

where $\lfloor \cdot \rfloor$ denotes rounding downwards to the closest integer and $\lceil \cdot \rceil$ denotes rounding upwards to the closest integer. Add both created nodes to the BB tree and go to step 1.

In order to illustrate the BB algorithm in a more intuitive way, a simple example is given in the following which is based on [CBD11]. Given the integer problem (2.45) denoted as \mathcal{O}_e as depicted below

$$\max_{\mathbf{z}} \quad E(\mathbf{z}) = 5z_1 - 2z_2 \quad (2.45a)$$

s. t.

$$-z_1 + 2z_2 \leq 5 \quad (2.45b)$$

$$3z_1 + 2z_2 \leq 19 \quad (2.45c)$$

$$z_1 + 3z_2 \geq 9 \quad (2.45d)$$

$$\mathbf{z} \in \mathbb{Z}_+^{2 \times 1} \quad (2.45e)$$

where the objective variables in \mathbf{z} are assumed to be non-negative integers. The goal of program (2.45) is to find the integer solution that maximizes the objective function $E(\mathbf{z})$ given linear inequalities (2.45b) to (2.45c). Note that problem (2.45) is a linear integer program due to the linear objective function $E(\mathbf{z})$. In Figure 2.4, the enumeration tree that reflects the BB solution procedure to solve problem (2.45) is displayed. Note that the lowest (best) upper bound on a subproblem is placed above the corresponding node, while the best lower bound on the original problem, i.e., \mathcal{O}_e , is placed below the nodes. The inequality constraints displayed at the edges correspond to the branching step of the above described BB algorithm where the solution space of a parent subproblem is partitioned into disjoint solution spaces of child subproblems in order to cut off regions of the solution set that are not of interest.

Initially, problem \mathcal{O}_e is continuously relaxed and solved. The best upper bound on problem \mathcal{O}_e is found to be $\bar{b} = 25.57$ and the best lower bound is set to $\underline{b} = -\infty$. The solution to the relaxed problem of \mathcal{O}_e , i.e., $\mathcal{O}_{cr, e}$, is given by $\mathbf{z} = (5.57, 1.14)$. Since this solution is non-integer, the branching proceeds.

Based on problem \mathcal{O}_e , two subproblems, denoted as \mathcal{O}_e^1 and \mathcal{O}_e^2 , are created by branching on variable z_1 as depicted in Figure 2.4. Solving the continuous relaxation of problem \mathcal{O}_e^1 denoted as $\mathcal{O}_{cr, e}^1$ yields that the problem is infeasible. Hence, no further exploration proceeding from this node is possible and the tree is pruned at this node. On the contrary, subproblem \mathcal{O}_e^2 is continuously relaxed and can be solved. The best upper bound on subproblem \mathcal{O}_e^2 is found to be $\bar{b}^2 = 22.33$, while the best lower bound on the original IP is still unchanged. The solution to problem $\mathcal{O}_{cr, e}^2$ is given by $\mathbf{z}^2 = (5, 4/3)$. Since not all components of \mathbf{z}^2 are integer values the branching procedure proceeds.

Based on subproblem \mathcal{O}_e^2 two child subproblems denoted as \mathcal{O}_e^3 and \mathcal{O}_e^4 are created by branching at z_2 according to the inequality constraints depicted in Figure 2.4. Solving the continuous relaxation of problem \mathcal{O}_e^3 , i.e., $\mathcal{O}_{cr, e}^3$, yields a best upper bound $\bar{b}^3 = 21$ on problem \mathcal{O}_e^3 . Since the solution of $\mathcal{O}_{cr, e}^3$ is integer, i.e., $\mathbf{z}^3 = (5, 2)$, it is also the solution to problem \mathcal{O}_e^3 and a candidate solution to the original problem. Hence, the best lower bound on problem \mathcal{O}_e has to be updated and set to $\underline{b} = 21$. Since the solution of problem \mathcal{O}_e^3 is already integer, no further branching has to be done starting from this node. On the contrary, solving for the continuous relaxation of subproblem \mathcal{O}_e^4 yields no feasible solution. Since no further exploration of the tree is possible starting from this node, the tree is pruned here.

Ultimately, the solution to problem \mathcal{O}_e corresponds to the solution of subproblem \mathcal{O}_e^3 and is given by $\mathbf{z} = \mathbf{z}^3 = (5, 2)$.

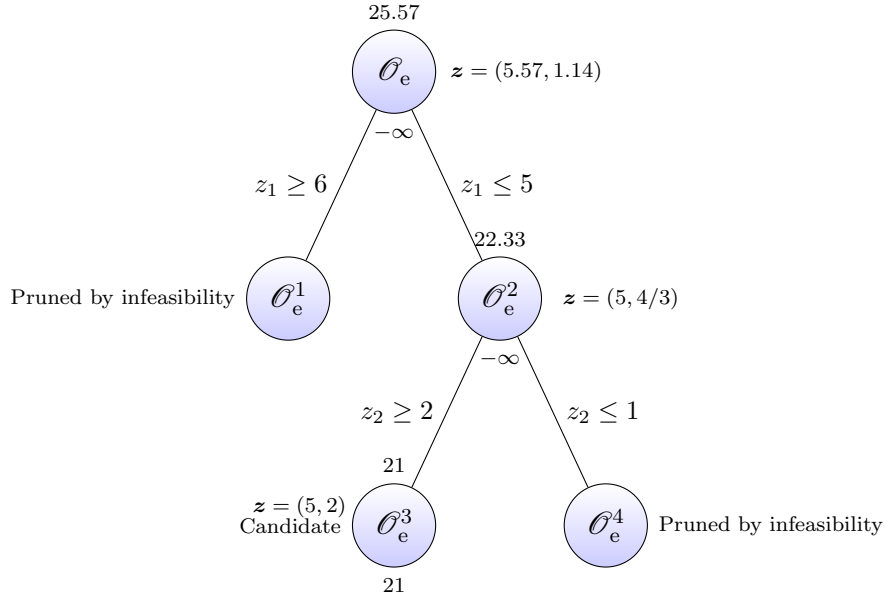


Figure 2.5. Branch-and-bound enumeration tree of problem (2.45).

As a generalization of the BB method, the BC algorithm is the state-of-the-art technique for solving integer and mixed integer problems and strongly builds on the ideas of the BB algorithm presented above. The essential difference between both methods is that the BC algorithm utilizes more sophisticated cuts in the branching step than the BB algorithm. In essence, the branching step of the BC algorithm makes use of advanced cut generation techniques based on the cutting-plane-approach and the group-theoretic-approach, respectively, and may involve adjusted modeling and preprocessing techniques [CBD11]. In commercial solvers, such as CPLEX [Flo95], Mosek [BV11] and Gurobi [Don], various methods of solving IPs and MIPs have been implemented, including the above mentioned BB and BC algorithms. Today, integer/mixed integer solvers have become quite powerful, although their performance critically depends on the used machine and their software framework. Nevertheless, problem instances of more than 100 integer variables are computationally tractable on average office machines.

Chapter 3

Integer Linear Programming for Graph Learning

In this chapter, ILP algorithms are developed based on SK, DK and GI-profile data in order to learn the pairwise genetic interaction classes according to [BJW⁺10] of the studied genes as well as the corresponding DAG/GI-network topology that reflects the mutual dependencies among all genes under study, i.e., the genetic interaction types/classes of all gene pairs. This chapter is based on the work published in [NPKT17], [NP16], [NP17] and [NP18].

In particular, the major contributions of this chapter are summarized in the following:

- In Section 3.1, the GENIE algorithm is presented where the estimation of the set of genetic interactions and the corresponding DAG/GI-network topology is done in a sequential fashion. First, the hierarchical relationship classes of [BJW⁺10] are learned based on the SK and DK data for each pair of genes under study. Since the genetic interaction types of different pairs of genes are highly coupled, as shown in Section 2.2.3, learning the hierarchical relationship classes for all gene pairs under investigation poses a highly coupled multi-class detection problem which is formulated as an ILP. Secondly, based on the learned genetic interaction classes of [BJW⁺10] a procedure to compute an estimate of the true DAG/GI-network topology, that is underlying the observed SK and DK data, is proposed.
- In Section 3.2, the GI-GENIE algorithm is presented. The proposed GI- GENIE algorithm combines multiple data types, i.e., SK and DK data with GI-profile data, in order to learn the set of hierarchical relationship classes of [BJW⁺10] and the corresponding DAG/GI-network topology jointly.
- In Section 3.3, the SEQSCA method is presented to compensate for the poor scalability, with respect to the number of genes under study, of the proposed GENIE and GI-GENIE algorithms, respectively. Based on the GENIE and GI-GENIE algorithm, respectively, the proposed sequential scalability technique yields an estimate of soft-decision interaction classes compared to the ones of [BJW⁺10] for large-scale gene networks. In particular, the SEQSCA method yields a GI-network topology estimate which is a fully connected graph of weighted edges. In this GI-network topology estimate, the empirical probability of two genes to

interact is described by the weight of the edge that links the associated genes with each other. Hence, the SEQSCA algorithm does not yield GI-network estimates that follow the terms of [BJW⁺10] and the associated data model \mathfrak{M}_B .

- In Section 3.4, the GENIE, GI-GENIE and SEQSCA algorithms are evaluated in terms of estimation performance for both simulated data and real data.

3.1 GENIE Algorithm

In this section, the GENIE algorithm, that learns the DAG/GI-network underlying the observed SK and DK data, is derived. First in Subsection 3.1.1, the estimation of the hierarchical relationship class representation of the underlying GI-network is formulated as an ILP. Secondly in Subsection 3.1.2, an algorithm to reconstruct the topology of the underlying DAG/GI-network based on its learned hierarchical relationship class representation from Subsection 3.1.1 is presented.

3.1.1 Hierarchical Relationship Class Learning

Let the SKs $R(i) \forall i \in \mathcal{G}$ and DKs $R(i, j) \forall \{i, j\} \in \mathcal{G} : j > i$ be given and the underlying DAG/GI-network \mathcal{D} be unknown. Assuming that the gene pair $\{i, j\}$ belongs to the hierarchical relationship class $k \in \mathcal{K}$ in DAG/GI-network \mathcal{D} , the mismatch between the actually observed DK phenotype $R(i, j)$ and the expected phenotype $\mu_k(i, j)$ under class k is quantified by the quadratic mismatch score $s_k(i, j)$ as defined in Eq. (3.1)

$$s_k(i, j) = (R(i, j) - \mu_k(i, j))^2, \quad k \in \mathcal{K} \quad \forall \{i, j\} \in \mathcal{G} : j > i. \quad (3.1)$$

Furthermore, let the class-selection variables¹ be defined as given below by Eq. (3.2)

$$\alpha_{k,i,j} = \begin{cases} 1 & \text{if } \{i, j\} \text{ are in class } k \\ 0 & \text{else} \end{cases} \quad k \in \mathcal{K}, \quad \forall \{i, j\} \in \mathcal{G} : j > i \quad (3.2)$$

where $\alpha_{k,i,j}$ indicates that genes $\{i, j\}$ are in class k in DAG \mathcal{D} . Note that given $\alpha_{k,i,j} = 1$, then all $\alpha_{k',i,j}$ for $k' \in \mathcal{K} \setminus \{k\}$ are 0, since according to conditions C_1 to C_5 of Table 2.1 a pair of genes in a DAG is always and exclusively classified to one class. Furthermore, it is sufficient and efficient to define the class selection variables $\alpha_{k,i,j}$ for

¹In a discrete optimization context the class-selection variables defined in (3.2) are denoted as SOS-1 type variables. However, for the sake of readability this optimization context based annotation is mostly omitted and the variables defined in (3.2) are referred to as class-selection variables.

gene pairs $\{i, j\} : \in \mathcal{G} : j > i$ only, since the gene pairs $\{i, j\}$ and $\{j, i\}$ are identical. It is important to remark that any DAG \mathcal{D} can be represented by a corresponding collection of class selection variables

$$A_{\mathcal{D}} = \bigcup_{\{i,j\} \in \mathcal{G} : j > i, k \in \mathcal{K}} \alpha_{k,i,j}. \quad (3.3)$$

There is a close relationship between the multi-categorical class indicator coefficients $\omega_{i,j}$ in Eq. (2.16) and the binary class selection variables $\alpha_{k,i,j}$ in Eq. (3.2) that is given by Eq. (3.4)

$$\alpha_{k,i,j} = \begin{cases} 1 & \text{if } \omega_{i,j} = k \\ 0 & \text{otherwise} \end{cases} \quad k \in \mathcal{K}, \{i, j\} \in \mathcal{G} : j > i. \quad (3.4)$$

In essence, for each pair of genes $\{i, j\} \in \mathcal{G} : j > i$ the multi-categorical class indicator coefficient $\omega_{i,j}$ is represented by a collection of five binary class selection variables $\{\alpha_{1,i,j}, \dots, \alpha_{5,i,j}\}$. Hence, the representation of DAG \mathcal{D} in the domain of hierarchical relationship classes of [BJW⁺10], i.e., $\Omega(\mathcal{D}) = \Omega_0$, uniquely corresponds to $A_{\mathcal{D}}$. Therefore, any reconstruction $\hat{\mathcal{D}}$ of DAG \mathcal{D} based on $A_{\mathcal{D}}$ can always be learned up to DAG-family-equivalence which is dictated by Ω_0 that uniquely corresponds to $A_{\mathcal{D}}$.

The problem of learning the set of class selection variables $A_{\mathcal{D}}$ corresponding to DAG/GI-network \mathcal{D} , that selects class k with the minimal mismatch score $s_k(i, j)$ for all pairs of genes $\{i, j\} \in \mathcal{G} : j > i$, has been formulated in Eqs. (3.5) as an ILP and is denoted as the GENIE-problem as stated below

$$\min_{\{\alpha_{k,i,j}\}} \sum_{\{i,j\} \in \mathcal{G} : j > i} \sum_{k \in \mathcal{K}} \alpha_{k,i,j} s_k(i, j) \quad (3.5a)$$

s. t.

$$\alpha_{k,i,j} \in \{0, 1\} \quad \forall \{i, j\} \in \mathcal{G} : j > i, \forall k \in \mathcal{K} \quad (3.5b)$$

$$\sum_{k \in \mathcal{K}} \alpha_{k,i,j} = 1, \quad \forall \{i, j\} \in \mathcal{G} : j > i \quad (3.5c)$$

$$\mathcal{L} \implies \text{class coupling constraints} \quad (3.5d)$$

where the solution to the GENIE-problem is denoted by

$$A_G = \bigcup_{\{i,j\} \in \mathcal{G} : j > i, k \in \mathcal{K}} \alpha_{G,k,i,j}. \quad (3.6)$$

Note that the GENIE-problem can be solved by BB and BC algorithms that are efficiently implemented in commercial solvers as CPLEX [Flo95], Gurobi [Don] and Mosek [BV11]. In particular, the optimization objective of the GENIE-problem as depicted by Eq. (3.5a) is to minimize the overall mismatch that originates from classifying all relevant gene pairs $\{i, j\}$ to specific classes of [BJW⁺10]. In other words,

the objective of the GENIE-problem is to find the set of class selection variables A_G that is associated with the lowest overall classification mismatch score S_{\min} as defined in Eq. (3.7) below

$$S_{\min} = \sum_{\{i,j\} \in \mathcal{G}: j > i} \sum_{k \in \mathcal{K}} \alpha_{G,k,i,j} s_k(i,j). \quad (3.7)$$

The constraint given by Eq. (3.5b) reflects the binary nature of the class selection variables $\alpha_{k,i,j}$. The multiple-choice-constraints stated by Eq. (3.5c) account for the fact that any pair of genes in a DAG under the terms of [BJW⁺10] belongs to one hierarchical relationship class at most, i.e., any pair of genes $\{i, j\}$ can be exclusively classified to only one class $k \in \mathcal{K}$. Furthermore, the set of constraints \mathcal{L} stated in Eq. (3.5d) contains a plethora of linear integer constraints that account for the logical coupling among the class indicator coefficients $\omega_{i,j}$, i.e., the class selection variables $\alpha_{k,i,j}$. By modeling the logical coupling among the class selection variables $\alpha_{k,i,j}$ as linear integer constraints aggregated in \mathcal{L} , the solution of the GENIE-problem is ensured to represent a DAG/GI-network which follows the topology restrictions given by the data model of [BJW⁺10] described by \mathfrak{M}_B . Hence, the learned set of class selection variables A_G is guaranteed to directly correspond to a set of class indicator coefficients $\Omega_G = \Omega(\mathcal{D}_G)$, which reflects DAG \mathcal{D}_G , that is an estimate of the true DAG/GI-network topology \mathcal{D} .

In order to identify all logical coupling constraints among the $\alpha_{k,i,j}$, contained in constraints set \mathcal{L} , it is sufficient to investigate the coupling among the class selection variables of all relevant triplets of genes $\{i, j, l\} \in \mathcal{G} : l > j > i$.

Definition 3.1.1. *The interaction set $\mathcal{I}^{(i)} \subset \mathcal{G}$ of gene $i \in \mathcal{G}$ defined by Eq. (3.8)*

$$\mathcal{I}^{(i)} = \left\{ l \mid \sum_{k \in \mathcal{K}} \alpha_{k,i,l} = 1, l \in \mathcal{G} : l > i \right\} \quad (3.8)$$

contains all genes l that are classified to some interaction class $k \in \mathcal{K}$ with respect to gene i .

Corollary 3.1.1. *For genes $\{i, j, l\} \in \mathcal{G} : l > j > i$, given that $\{j, l\} \in \mathcal{I}^{(i)}$, then the classification of gene pair $\{j, l\}$ is subject to logical coupling, i.e., pair $\{j, l\}$ is restricted to belong to classes $\mathcal{K}' \subseteq \mathcal{K}$ that is determined by the interaction classes k and k' of pairs $\{i, j\}$ and $\{i, l\}$, respectively.*

According to Corollary 3.1.1, a triplet of genes $\{i, j, l\} \in \mathcal{G} : l > j > i$ can be decomposed into three pairs, i.e., $\{i, j, l\} \implies \{i, j\}, \{i, l\}, \{j, l\}$, where the classification of the pair $\{j, l\}$ is restricted to a subset of classes \mathcal{K}' given that pairs $\{i, j\}$ and $\{i, l\}$ are already classified. It is important to remark that, given the classification of pairs $\{i, j\}$

and $\{j, l\}$, the classification of pair $\{i, l\}$ is restricted to a subset of classes $\mathcal{K}'' \subseteq \mathcal{K}$ as well. In essence, given the pairwise decomposition of a triplet of genes, restrictions in classifying one of those pairs occur if the remaining two pairs are already classified.

Based on the pairwise decomposition $\{i, j\}, \{i, l\}, \{j, l\}$ of any relevant triplet of genes $\{i, j, l\} \in \mathcal{G} : l > j > i$, any algorithm to identify the coupling constraints among the class selection variables defined in Eq. (3.2) ought to implement the following blueprint that is described on a high-level scale:

- (a) Assume that the gene pairs $\{i, j\}$ and $\{i, l\}$ are classified to classes $k \in \mathcal{K}$ and $k' \in \mathcal{K}$, respectively. In the domain of class selection variables this assumption corresponds to $\alpha_{k,i,j} = 1$ and $\alpha_{k',i,l} = 1$.
- (b) Given the assumptions in step (a), identify the set of classes $\mathcal{K}' \subseteq \mathcal{K}$ that the gene pair $\{j, l\}$ is restricted to
- (c) Repeat steps (a) and (b) for all combinations of classes $\{k, k'\} \in \mathcal{K}$, i.e., for all $\alpha_{k,i,j} = 1$ and $\alpha_{k',i,l} = 1$.

Under the assumptions that $\alpha_{k,i,j} = 1$ and $\alpha_{k',i,l} = 1$ with $k, k' \in \mathcal{K}$, the identification of the set of classes $\mathcal{K}' \subseteq \mathcal{K}$, that gene pair $\{j, l\}$ is restricted to, poses a critical and challenging part of the above proposed high-level procedure. In the following an algorithm is derived that accounts for the above described high-level routine (a) to (c) by making use of *sub-genetic-interactions-maps* (SMAP)s as introduced by Definition 3.1.2.

Definition 3.1.2. Let DAG $\mathcal{D} = (\mathcal{V}_{\mathcal{D}}, \mathcal{E}_{\mathcal{D}})$ and a subgraph $\mathcal{S} = (\mathcal{V}_{\mathcal{S}}, \mathcal{E}_{\mathcal{S}})$ in \mathcal{D} be given, such that $\mathcal{V}_{\mathcal{S}} \subseteq \mathcal{V}_{\mathcal{D}} \setminus \{R\}$ and $\mathcal{E}_{\mathcal{S}} \subseteq \mathcal{E}_{\mathcal{D}}$. Furthermore, the set of “inward” genes $\mathcal{N}_{\mathcal{S},in}$ of subgraph \mathcal{S} and the set of “outward” genes $\mathcal{N}_{\mathcal{S},out}$ of graph \mathcal{S} are given by:

$$\mathcal{N}_{\mathcal{S},in} = \left\{ g_i \mid g_i \in \mathcal{V}_{\mathcal{S}}, \{g_e, g_i\} \in \mathcal{E}_{\mathcal{D}} \text{ for } \{g_e\} \in \mathcal{V}_{\mathcal{D}} \setminus \mathcal{V}_{\mathcal{S}} \right\} \quad (3.9)$$

$$\mathcal{N}_{\mathcal{S},out} = \left\{ g_i \mid g_i \in \mathcal{V}_{\mathcal{S}}, \{g_i, g_e\} \in \mathcal{E}_{\mathcal{D}} \text{ for } \{g_e\} \in \mathcal{V}_{\mathcal{D}} \setminus \mathcal{V}_{\mathcal{S}} \right\} \quad (3.10)$$

Then subgraph \mathcal{S} in DAG \mathcal{D} is a SMAP if the following conditions are true:

1 \mathcal{S} is acyclic and directed

2 $\forall g_i \in \mathcal{V}_{\mathcal{S}}$ there is at least one path $P_{\tau}^{(i)}$ that starts at g_i and terminates in the reporter node $R \implies \mathcal{N}_{\mathcal{S},out} \neq \emptyset$

3 for any path P in DAG \mathcal{D} traversing \mathcal{S} : $\exists g_i \in \mathcal{N}_{\mathcal{S},in}, \exists g_j \in \mathcal{N}_{\mathcal{S},out} : g_i, g_j \in V(P)$

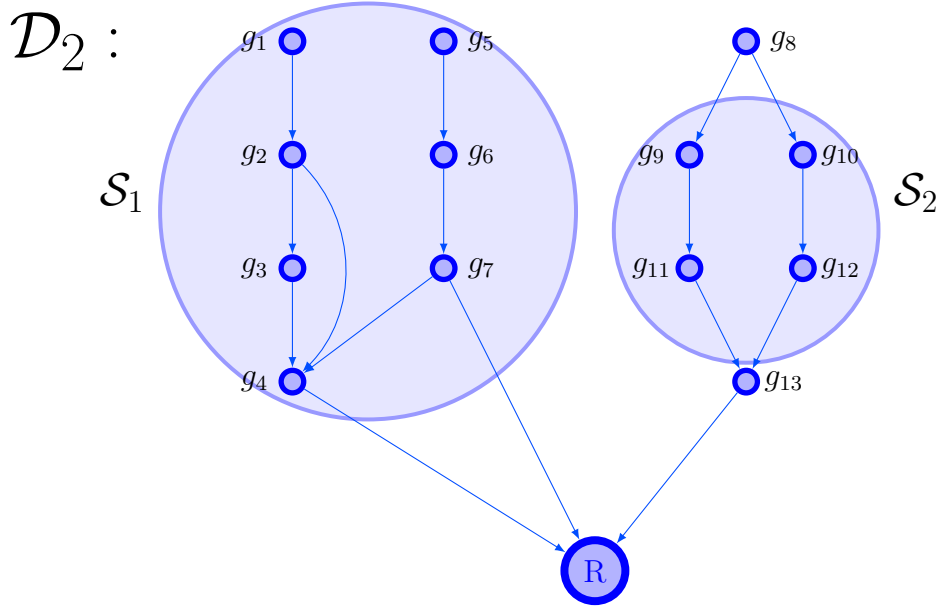


Figure 3.1. DAG \mathcal{D}_2 with the set of genes $\mathcal{G}_{\mathcal{D}_2} = \{g_1, \dots, g_{13}\}$ and the set of edges $\mathcal{E}_{\mathcal{D}_2}$ that reflects the displayed topology. The SMAPs $\mathcal{S}_1 = \{\mathcal{V}_{\mathcal{S}_1}, \mathcal{E}_{\mathcal{S}_1}\}$ and $\mathcal{S}_2 = \{\mathcal{V}_{\mathcal{S}_2}, \mathcal{E}_{\mathcal{S}_2}\}$ are highlighted where $\mathcal{V}_{\mathcal{S}_1} = \{g_1, \dots, g_7\}$, $\mathcal{V}_{\mathcal{S}_2} = \{g_9, \dots, g_{12}\}$ and sets of edges $\mathcal{E}_{\mathcal{S}_1}$ and $\mathcal{E}_{\mathcal{S}_2}$, respectively, that reflect the corresponding displayed topologies.

Excluding the reporter node R , any subgraph in DAG \mathcal{D} can be represented by a SMAP \mathcal{S} according to Definition 3.1.2. To convey a better understanding of the conceptual idea of a SMAP as defined in Definition 3.1.2, in Example 3.1.1 the condensed representation of a DAG based on SMAPs is illustrated.

Example 3.1.1. *Without any loss of information, DAG \mathcal{D}_2 as depicted in Figure 3.1 can be transformed into a reduced/condensed representation using SMAPs \mathcal{S}_1 and \mathcal{S}_2 that are highlighted in Figure 3.1. In particular, the following holds for SMAP \mathcal{S}_1 :*

- \mathcal{S}_1 is acyclic and directed
- $\mathcal{N}_{\mathcal{S}_1, \text{out}} = \{g_4, g_7\} \neq \emptyset$ and furthermore, for each gene in \mathcal{S}_1 there is a path to the reporter node R
- since there is no path traversing \mathcal{S}_1 , condition 3 in Definition 3.1.2 is automatically fulfilled

For SMAP \mathcal{S}_2 the following is true:

- \mathcal{S}_2 is acyclic and directed
- $\mathcal{N}_{\mathcal{S}_2, \text{in}} = \{g_9, g_{10}\}$
- $\mathcal{N}_{\mathcal{S}_2, \text{out}} = \{g_{11}, g_{12}\} \neq \emptyset$ and moreover, for each gene in \mathcal{S}_2 there is a path to the reporter node R

$\mathcal{D}_2 :$

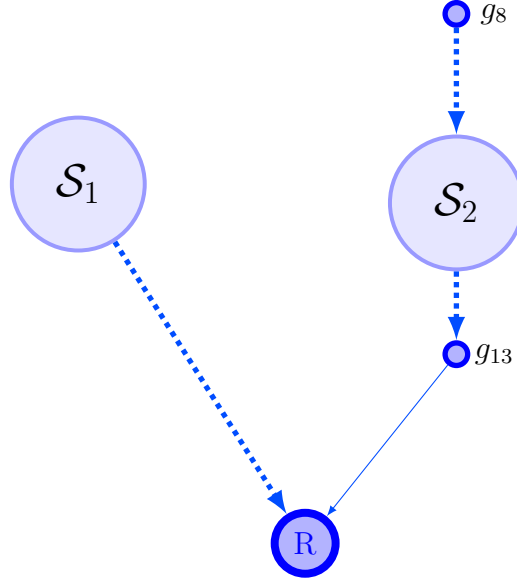


Figure 3.2. DAG \mathcal{D}_2 in a condensed representation based on the SMAPS $\mathcal{S}_1 = \{\mathcal{V}_{\mathcal{S}_1}, \mathcal{E}_{\mathcal{S}_1}\}$ and $\mathcal{S}_2 = \{\mathcal{V}_{\mathcal{S}_2}, \mathcal{E}_{\mathcal{S}_2}\}$ from Figure 3.1. Note that large dashed edges are used to describe the interaction between a node, i.e., a gene or the reporter level, and a SMAP. This notation takes account for the fact that there can be multiple edges between a gene and the "inward" and "outward" genes of a SMAP as stated in Definition 3.1.2.

- *there are two paths traversing \mathcal{S}_2 and terminating in the reporter node R : $P_1^{(g_8)} = g_8, g_9, g_{11}, g_{13}, R$ and $P_2^{(g_8)} = g_8, g_{10}, g_{12}, g_{13}, R$. For path $P_1^{(g_8)}$, genes $g_9 \in \mathcal{N}_{\mathcal{S}_2, in}$, $g_{11} \in \mathcal{N}_{\mathcal{S}_2, out}$ are in $V(P_1^{(g_8)})$. For path $P_2^{(g_8)}$, genes $g_{10} \in \mathcal{N}_{\mathcal{S}_2, in}$, $g_{12} \in \mathcal{N}_{\mathcal{S}_2, out}$ are in $V(P_2^{(g_8)})$. Hence, condition 3 in Definition 3.1.2 is fulfilled.*

As illustrated in Example 3.1.1, DAG \mathcal{D}_2 of Figure 3.1 can be represented in a reduced form based on the SMAPs \mathcal{S}_1 and \mathcal{S}_2 as displayed in Figure 3.2. In essence, the representation of any DAG \mathcal{D} can be reduced/simplified to different degrees using SMAPs. Note that the most reduced representation of a DAG only consists of one large SMAP and the reporter level R .

Incorporating the considerations stated in (a) to (c) as well as the concept of SMAPs as introduced in Definition 3.1.2, the procedure stated as Algorithm 3 yields the full set of class coupling constraints \mathcal{L} as stated in Eq. (3.5d). In the following, the set of coupling constraints for triplets $\{i, j, l\} \in \mathcal{G} : l > j > i$ is exemplarily derived according to Algorithm 3 under the assumptions that $k = 1$ and for all $k' \in \mathcal{K}$. Hence, one iteration over k , i.e., of the outer *for*-loop, is computed that yields the set \mathcal{L}_1

Algorithm 3 Coupling Constraints Identification

```
1: for  $k \in \mathcal{K}$  do
2:   for  $\{i, j, l\} \in \mathcal{G} : l > j > i$  do
3:     decompose triplet  $\{i, j, l\}$  into pairs  $\{i, j\}$ ,  $\{i, l\}$ ,  $\{j, l\}$ 
4:     for  $k' \in \mathcal{K}$  do
5:       set  $\alpha_{k,i,j} = 1$ ,  $\alpha_{k',i,l} = 1$ , i.e., classify pairs  $\{i, j\}$  and  $\{i, l\}$ 
        to classes  $k$  and  $k'$ , respectively
6:
7:       develop relevance DAG  $\mathcal{D}_r$  based on SMAPs
8:
9:       conclude  $\mathcal{K}' \subseteq \mathcal{K}$  from particular relevance DAG
10:
11:      translate  $\mathcal{K}'$  and assumptions to class selection variable domain  $\alpha$ 
      return  $\Rightarrow \mathcal{L}_k$ : all coupling rules given that  $\{i, j\}$  in class  $k$ , i.e.,  $\alpha_{k,i,j} = 1$ 
return  $\mathcal{L} = \bigcup_{k \in \mathcal{K}} \{\mathcal{L}_k\}$ 
```

which contains all coupling rules of the gene pairs $\{i, j\}$, $\{i, l\}$ and $\{j, l\}$ in the case of $\alpha_{1,i,j} = 1$. Following line 2 of Algorithm 3, each triplet of genes $\{i, j, l\} \in \mathcal{G} : l > j > i$ is decomposed into pairs $\{i, j\}$, $\{i, l\}$ and $\{j, l\}$. Then for each decomposed triplet of genes, the following steps are performed:

step 1: Given that $k = 1$ and $k' = 1$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{1,i,l} = 1$. Furthermore, DAG $\mathcal{D}_{r,1}$ displayed in Figure 3.3 is composed of genes i and j as well as of SMAPs $\mathcal{S}_{r,1}$ and $\mathcal{S}_{r,2}$. DAG $\mathcal{D}_{r,1}$ is denoted as the *relevance DAG* for the given assumptions, since it only models the topology that is relevant for identifying the restricted set of classes \mathcal{K}' which pair $\{j, l\}$ is allowed to be classified to. In DAG $\mathcal{D}_{r,1}$, it is obvious that gene pair $\{i, j\}$ is in class $k = 1$, i.e., $\alpha_{1,i,j} = 1$. Furthermore, any gene l that fulfills the assumption that $\alpha_{1,i,l} = 1$, must be located either in SMAP $\mathcal{S}_{r,1}$ or $\mathcal{S}_{r,2}$. In the case of gene l being located in $\mathcal{S}_{r,1}$, gene pair $\{j, l\}$ is in class $k'' = 2$. On the contrary, gene pair $\{j, l\}$ is in class $k'' = 1$ if gene l is located in SMAP $\mathcal{S}_{r,2}$. Hence, the set of classes, that gene pair $\{j, l\}$ is restricted to belong to, is given by $\mathcal{K}' = \{1, 2\}$.

step 2: Given that $k = 1$ and $k' = 2$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{2,i,l} = 1$. Furthermore, DAG $\mathcal{D}_{r,2}$ displayed in Figure 3.3 is composed of genes i and j as well as of SMAP $\mathcal{S}_{r,3}$. DAG $\mathcal{D}_{r,2}$ is denoted as the *relevance DAG* for the assumptions made, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{2,i,l} = 1$. In DAG $\mathcal{D}_{r,2}$, it is obvious that gene pair $\{i, j\}$ is in class $k = 1$, i.e., $\alpha_{1,i,j} = 1$. Moreover, any gene l that fulfills the assumption that $\alpha_{2,i,l} = 1$, must be located in SMAP $\mathcal{S}_{r,3}$. Thus, any gene l that is located in $\mathcal{S}_{r,3}$ must be in class $k'' = 2$ with gene j , i.e., $\alpha_{2,j,l} = 1$. Hence, the set of classes, that gene pair $\{j, l\}$ is restricted to belong to, is given by $\mathcal{K}' = \{2\}$.

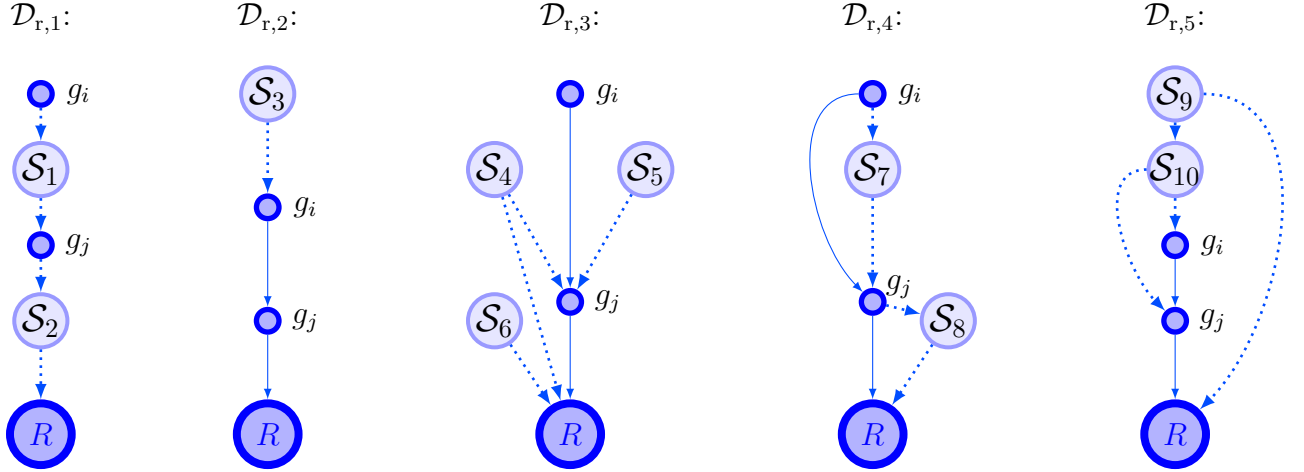


Figure 3.3. Relevance DAGs to identify the class coupling constraints under the assumptions that $\alpha_{1,i,j} = 1$ and $\alpha_{k',i,l} = 1$ for $k' \in \mathcal{K}$. $\mathcal{D}_{r,1}$: relevance DAG for $\alpha_{1,i,j} = 1$ and $\alpha_{1,i,l} = 1$; $\mathcal{D}_{r,2}$: relevance DAG for $\alpha_{1,i,j} = 1$ and $\alpha_{2,i,l} = 1$; $\mathcal{D}_{r,3}$: relevance DAG for $\alpha_{1,i,j} = 1$ and $\alpha_{3,i,l} = 1$; $\mathcal{D}_{r,4}$: relevance DAG for $\alpha_{1,i,j} = 1$ and $\alpha_{4,i,l} = 1$; $\mathcal{D}_{r,5}$: relevance DAG for $\alpha_{1,i,j} = 1$ and $\alpha_{5,i,l} = 1$

- step 3: Given that $k = 1$ and $k' = 3$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{3,i,l} = 1$. Furthermore, DAG $\mathcal{D}_{r,3}$ displayed in Figure 3.3 is composed of genes i and j as well as of SMAPs $\mathcal{S}_{r,4}$, $\mathcal{S}_{r,5}$ and $\mathcal{S}_{r,6}$. In relevance DAG $\mathcal{D}_{r,3}$, it is obvious again that gene pair $\{i, j\}$ is in class $k = 1$, i.e., $\alpha_{1,i,j} = 1$. Any gene l that fulfills the assumption that $\alpha_{3,i,l} = 1$, must be either located in SMAP $\mathcal{S}_{r,4}$, $\mathcal{S}_{r,5}$ or $\mathcal{S}_{r,6}$. Thus, it follows that the gene pair $\{j, l\}$ is restricted to belong $\mathcal{K}' = \{2, 3, 5\}$.
- step 4: Given that $k = 1$ and $k' = 4$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{4,i,l} = 1$. Furthermore, DAG $\mathcal{D}_{r,4}$ displayed in Figure 3.3 is composed of genes i and j as well as of SMAPs $\mathcal{S}_{r,7}$ and $\mathcal{S}_{r,8}$. In relevance DAG $\mathcal{D}_{r,4}$, gene pair $\{i, j\}$ is in class $k = 1$, i.e., $\alpha_{1,i,j} = 1$. Furthermore, any gene l that fulfills the assumption that $\alpha_{4,i,l} = 1$, must be either located in SMAP $\mathcal{S}_{r,7}$ or SMAP $\mathcal{S}_{r,8}$. Thus, it follows that the gene pair $\{j, l\}$ is restricted to belong classes $\mathcal{K}' = \{2, 4\}$.
- step 5: Given that $k = 1$ and $k' = 5$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{5,i,l} = 1$. Furthermore, DAG $\mathcal{D}_{r,5}$ displayed in Figure 3.3 is composed of genes i and j as well as of SMAPs $\mathcal{S}_{r,9}$ and $\mathcal{S}_{r,10}$. In relevance DAG $\mathcal{D}_{r,5}$, gene pair $\{i, j\}$ is in class $k = 1$, i.e., $\alpha_{1,i,j} = 1$. Furthermore, any gene l that fulfills the assumption that $\alpha_{5,i,l} = 1$, must be either located in SMAP $\mathcal{S}_{r,9}$ or SMAP $\mathcal{S}_{r,10}$. Thus, it follows that the gene pair $\{j, l\}$ is restricted to belong classes $\mathcal{K}' = \{2, 5\}$.

In the following, the coupling rules of steps 1. to 5. as described above are translated into integer linear inequalities in terms of the selection variables $\alpha_{k,i,j}$ and given by

constraints (3.11a) to (3.11e) of set \mathcal{L}_1 as defined in Eq. (3.11) by:

$$\mathcal{L}_1 = \left\{ \begin{array}{ll} \alpha_{1,j,l} + \alpha_{2,j,l} \geq \alpha_{1,i,j} + \alpha_{1,i,l} - 1 & (3.11a) \\ \alpha_{2,j,l} \geq \alpha_{1,i,j} + \alpha_{2,i,l} - 1 & (3.11b) \\ \alpha_{2,j,l} + \alpha_{3,j,l} + \alpha_{5,j,l} \geq \alpha_{1,i,j} + \alpha_{3,i,l} - 1 & (3.11c) \\ \alpha_{2,j,l} + \alpha_{4,j,l} \geq \alpha_{1,i,j} + \alpha_{4,i,l} - 1 & (3.11d) \\ \alpha_{5,j,l} + \alpha_{2,j,l} \geq \alpha_{1,i,j} + \alpha_{5,i,l} - 1 & (3.11e) \\ \forall i, j, l \in \mathcal{G} : l > j > i \end{array} \right.$$

The coupling rules derived in step 1. are modeled by the integer linear constraint in Eq. (3.11a). In particular, given that pair $\{i, j\}$ is in class $k = 1$ and pair $\{i, l\}$ belongs to class $k' = 1$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{1,i,l} = 1$, the RHS of Eq. (3.11a) amounts to 1 which enforces the LHS of Eq. (3.11a) to amount to 1 as well. Note that due to the multiple-choice constraint in Eq. (3.5c) of the GENIE-problem, only one $\alpha_{k,i,j}$, $k \in \mathcal{K}$ can be non-zero for each pair of genes $\{i, j\} \in \mathcal{G} : j > i$. Hence, either $\alpha_{1,j,l} = 1$ or $\alpha_{2,j,l} = 1$. In other words, gene pair $\{j, l\}$ can only be classified to classes $k'' \in \mathcal{K}' = \{1, 2\}$. On the other hand, given that $\alpha_{1,i,j} + \alpha_{1,i,l} \neq 2$ the constraint in Eq. (3.11a) is always fulfilled, i.e., no restrictions in classifying gene pair $\{j, l\}$ originate from Eq. (3.11a). In this case $\alpha_{1,i,j} + \alpha_{1,i,l} \neq 2$, the RHS of Eq. (3.11a) is less than 1, i.e., 0 or -1 . Since the LHS of Eq. (3.11a) is always greater or equal than 0, the inequality is always fulfilled.

The coupling rules derived in step 2. are modeled by the integer linear constraint in Eq. (3.11b). According to step 2., given that pair $\{i, j\}$ is in class $k = 1$ and pair $\{i, l\}$ belongs to class $k' = 2$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{2,i,l} = 1$, the RHS of Eq. (3.11b) amounts to 1 which enforces the LHS of Eq. (3.11b) to amount to 1 as well. Thus, $\alpha_{2,j,l} = 1$, i.e., gene pair $\{j, l\}$ must be classified to class $k'' = 2$ under the assumptions made in step 2.. On the contrary, given that $\alpha_{1,i,j} + \alpha_{2,i,l} \neq 2$, then Eq. (3.11b) is always fulfilled, i.e., no restrictions in classifying gene pair $\{j, l\}$ originate from Eq. (3.11b). To see this, the RHS of Eq. (3.11b) is less than 1, i.e., 0 or -1 and the LHS of Eq. (3.11b) is always greater or equal to 0. Hence, the LHS of Eq. (3.11b) is always greater or equal to the RHS of Eq. (3.11b), irrespectively of the choice of $\alpha_{2,j,l}$.

By constraint Eq. (3.11c), the coupling rules derived in step 3. are modeled as an integer linear inequality in terms of the class selection variables $\alpha_{k,i,j}$. In particular, given that pair $\{i, j\}$ is in class $k = 1$ and pair $\{i, l\}$ belongs to class $k' = 3$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{3,i,l} = 1$, the RHS of Eq. (3.11c) amounts to 1 which enforces the LHS of Eq. (3.11c) to amount to 1 as well. Again, note that due to the multiple-choice constraint in Eq. (3.5c) of the GENIE-problem, only one $\alpha_{k,i,j}$, $k \in \mathcal{K}$ can be non-zero for each pair

of genes $\{i, j\} \in \mathcal{G} : j > i$. Consequently, either $\alpha_{2,j,l} = 1$, $\alpha_{3,j,l} = 1$ or $\alpha_{5,j,l} = 1$. In essence, gene pair $\{j, l\}$ can only be classified to classes $k'' \in \mathcal{K}' = \{2, 3, 5\}$. On the contrary, assuming that $\alpha_{1,i,j} + \alpha_{3,i,l} \neq 2$, then Eq. (3.11c) is always fulfilled, i.e., no restrictions in classifying gene pair $\{j, l\}$ originate from Eq. (3.11c). In the case of $\alpha_{1,i,j} + \alpha_{3,i,l} \neq 2$, the RHS of Eq. (3.11c) is less than 1, i.e., 0 or -1 . Since the LHS of Eq. (3.11c) is always greater or equal than 0, the inequality is always fulfilled.

According to constraint Eq. (3.11d), the coupling rules derived in step 4. are modeled as an integer linear inequality in terms of the class selection variables $\alpha_{k,i,j}$. Given that pair $\{i, j\}$ is in class $k = 1$ and pair $\{i, l\}$ belongs to class $k' = 4$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{4,i,l} = 1$, the RHS of Eq. (3.11d) amounts to 1 which enforces the LHS of Eq. (3.11d) to amount to 1 as well. Due to the multiple-choice constraint in Eq. (3.5c) of the GENIE-problem, either $\alpha_{2,j,l} = 1$ or $\alpha_{4,j,l} = 1$. Thus, gene pair $\{j, l\}$ can only be classified to classes $k'' \in \mathcal{K}' = \{2, 4\}$. In contrast to that, given that $\alpha_{1,i,j} + \alpha_{4,i,l} \neq 2$, then Eq. (3.11d) is always fulfilled and no classification restrictions follow from Eq. (3.11d) for gene pair $\{j, l\}$. Under the assumption that $\alpha_{1,i,j} + \alpha_{4,i,l} \neq 2$, the RHS of Eq. (3.11d) is less than 1, i.e., 0 or -1 . Since the LHS of Eq. (3.11d) is always greater or equal than 0, the inequality is always fulfilled.

Finally, the coupling rules derived in step 5. are modeled by the integer linear constraint in Eq. (3.11e). According to step 5., given that pair $\{i, j\}$ is in class $k = 1$ and pair $\{i, l\}$ belongs to class $k' = 5$, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{5,i,l} = 1$, the RHS of Eq. (3.11e) amounts to 1 which enforces the LHS of Eq. (3.11e) to amount to 1 as well. Thus, either $\alpha_{2,j,l} = 1$ or $\alpha_{5,j,l} = 1$ must hold, i.e., gene pair $\{j, l\}$ must be classified to classes $k'' \in \mathcal{K}' = \{2, 5\}$. Reversely, given that $\alpha_{1,i,j} + \alpha_{5,i,l} \neq 2$, then Eq. (3.11e) is always fulfilled, i.e., no restrictions in classifying gene pair $\{j, l\}$ originate from Eq. (3.11e). This follows, since the RHS of Eq. (3.11e) is less than 1, i.e., 0 or -1 and the LHS of Eq. (3.11e) is always greater or equal to 0. Hence, the LHS of Eq. (3.11e) is always greater or equal to the RHS of Eq. (3.11e), irrespectively of the choice of $\alpha_{2,j,l}$ and $\alpha_{5,j,l}$.

In order to obtain the full set of class coupling constraints \mathcal{L} stated by Eq. (3.5d) of the GENIE-problem, Algorithm 3 has to be performed for all $k \in \mathcal{K}$, for all $k' \in \mathcal{K}$ and for all triplets of genes $\{i, j, l\} \in \mathcal{G} : l > j > i$ which yields the sets of class coupling rules \mathcal{L}_1 , \mathcal{L}_2 , \mathcal{L}_3 , \mathcal{L}_4 and \mathcal{L}_5 . Consequently, the full set of class coupling constraints \mathcal{L} is obtained as

$$\mathcal{L} = \bigcup_{k \in \mathcal{K}} \{\mathcal{L}_k\}. \quad (3.12)$$

The integer linear inequalities, that sets \mathcal{L}_2 , \mathcal{L}_3 , \mathcal{L}_4 and \mathcal{L}_5 are composed of, are given in Appendix B. It is important to remark, that one considerable advantage of the specific

formulation of the GENIE-problem as stated in Eq. (3.5) is its ability to incorporate prior knowledge into the class learning procedure. For instance, assume that for two specific genes i_0 and j_0 of the set of genes under study, i.e., \mathcal{G} , it is known that those genes do not interact with each. This prior knowledge can be easily incorporated into the proposed class learning procedure by adding the constraint

$$\alpha_{3,i_0,j_0} = 1 \quad \text{for } \{i_0, j_0\} \in \mathcal{G} : j_0 > i_0 \quad (3.13)$$

to the GENIE-problem that is stated in Eq. (3.5).

3.1.2 Graph Topology Reconstruction

In this subsection, an algorithm to learn the structure/topology $\mathcal{E}_{\mathcal{D}}$ of DAG \mathcal{D} , which is underlying the observed SK and DK data of the set of genes under study, is proposed. Based on the set of learned class selection variables A_G that directly corresponds to a set of class indicator coefficients $\Omega_G = \Omega(\mathcal{D}_G)$, a topology estimate $\mathcal{E}_{\mathcal{D},G}$ of the true DAG topology $\mathcal{E}_{\mathcal{D}}$ is learned in an iterative fashion. According to Lemma 2.2.1, DAG $\mathcal{D}_G = (\mathcal{G} \cup \{R\}, \mathcal{E}_{\mathcal{D},G})$, which is the GENIE-estimate of the true DAG \mathcal{D} , can only be learned up to DAG-family-equivalence, i.e., there might be multiple DAGs represented by A_G/Ω_G . Since it is known in biology that genetic interactions are generally scarce, the learned topology $\mathcal{E}_{\mathcal{D},G}$ should be least complex, i.e., having at most as many edges as needed in order to be represented by A_G/Ω_G . Consequently, the learned DAG \mathcal{D}_G is the sparsest DAG of its DAG-family \mathcal{F}_{Ω_G} , i.e., DAG \mathcal{D}_G has the smallest number of edges of all DAGs that belong to DAG-family \mathcal{F}_{Ω_G} .

In order to detect whether there is an edge between two genes $\{i, j\} \in \mathcal{G} : j > i$ in the GENIE-estimate \mathcal{D}_G of the true DAG \mathcal{D} , conditions E_1 to E_4 in Table 3.1 have to be evaluated. Given that one of the four mutually exclusive conditions E_1 to E_4 is true, then there is an edge in DAG \mathcal{D}_G between genes i and j . On the contrary, assuming that all conditions in Table 3.1 are false for a specific pair of genes, then there is no edge between those genes in DAG \mathcal{D}_G . It is important to remark that for the purpose of a compact and intuitive notation of the conditions in Table 3.1, the set of learned class selection variables A_G is redundantly expanded to contain class selection variables $\alpha_{G,k,i,j}$ for all pairs $\{i, j\} \in \mathcal{G}$. In particular, the redundantly expanded set of learned class selection variables \tilde{A}_G is given by Eq. (3.14)

$$\tilde{A}_G = \bigcup_{\{i,j\} \in \mathcal{G}, k \in \mathcal{K}} \tilde{\alpha}_{G,k,i,j} \quad (3.14)$$

with the redundantly expanded class selection variables $\tilde{\alpha}_{G,k,i,j}$ defined according to Eq. (3.15)

$$\tilde{\alpha}_{G,k,i,j} = \begin{cases} \alpha_{G,k,i,j} & \text{if } j > i \\ \alpha_{G,f(k),j,i} & \text{otherwise} \end{cases} \quad \forall \{i,j\} \in \mathcal{G}, \forall k \in \mathcal{K} \quad (3.15)$$

and the class mapping function $f(\cdot)$, which makes use of the symmetry properties of the hierarchical relationship classes of [BJW⁺10], is defined according to Eq. (3.16)

$$f(k) = \begin{cases} 2 & \text{if } k = 1 \\ 1 & \text{if } k = 2 \\ 5 & \text{if } k = 4 \\ 4 & \text{if } k = 5 \\ 3 & \text{otherwise} \end{cases} \quad \text{for } k \in \mathcal{K}. \quad (3.16)$$

The principal idea behind conditions E₁ to E₄ in Table 3.1 is based on the knowledge that the learned class selection variables in \tilde{A}_G , i.e. the corresponding set of relationship classes according to [BJW⁺10], contain hierarchical information about the genes in \mathcal{D}_G . Thus, the conditions in Table 3.1 basically describe an ordering procedure with respect to the learned class selection variables \tilde{A}_G .

As stated by Eq. (3.17a) of condition E₁ in Table 3.1, there is an edge directed from gene i to gene j in the GENIE estimate \mathcal{D}_G of the true DAG \mathcal{D} , given that the condition described by Eq. (3.17a) is true. In particular, assume that the gene pair $\{i,j\} \in \mathcal{G}$: $j > i$ has been classified to class $k = 1$ by the solution of the GENIE-problem, i.e., $\tilde{\alpha}_{G,1,i,j} = 1$. Furthermore, assume that there is no gene $l \in \mathcal{G} \setminus \{i,j\}$ for which $\tilde{\alpha}_{G,1,i,l} = 1$ and $\tilde{\alpha}_{G,2,j,l} = 1$ holds. Then parts EP_{1,1} and EP_{1,2} of Eq. (3.17a) are true which yields that condition E₁ is true. In this case, conditions E₂ to E₄ are automatically false due to the multiple-choice constraint in Eq. (3.5c) of the GENIE-problem which enforces that any pair of genes $\{i,j\}$ in a DAG exclusively belongs to one class. Note that part EP_{1,2} being false reflects an ordering constellation that directly contradicts the existence of an edge from gene i to gene j , i.e., $\{i,j\}$. To see this, the following argument is presented:

- $\tilde{\alpha}_{G,1,i,j} = 1$ implies that every path $P_\tau^{(i)}$ from gene i to the reporter node R traverses gene j
- $\tilde{\alpha}_{G,1,i,l} = 1$ implies that every path $P_\tau^{(i)}$ from gene i to the reporter node R traverses gene l as well
- $\tilde{\alpha}_{G,2,j,l} = 1$ implies that every path $P_\zeta^{(l)}$ from gene l to the reporter node R traverses gene j . Hence, any path from gene i to gene j traverses gene l

As a consequence, genes i and j cannot be neighbours in DAG \mathcal{D}_G , i.e., there cannot be a directed edge $\{i, j\}$ in DAG \mathcal{D}_G linking genes i and j .

$$\begin{aligned} E_1 : \quad & \text{if “true”} \longrightarrow \text{directed edge } \{i, j\} \in \mathcal{E}_G \\ & \left(\underbrace{\tilde{\alpha}_{G,1,i,j} = 1}_{=EP_{1,1}} \right) \wedge \left(\underbrace{\left(\nexists \{l\} \in \mathcal{G} \setminus \{i, j\} : \tilde{\alpha}_{G,1,i,l} = 1 \wedge \tilde{\alpha}_{G,2,j,l} = 1 \right)}_{=EP_{1,2}} \right) \end{aligned} \quad (3.17a)$$

$$\begin{aligned} E_2 : \quad & \text{if “true”} \longrightarrow \text{directed edge } \{i, j\} \in \mathcal{E}_G \\ & \left(\underbrace{\tilde{\alpha}_{G,4,i,j} = 1}_{=EP_{2,1}} \right) \wedge \left(\nexists \{l\} \in \mathcal{G} \setminus \{i, j\} : \right. \\ & \left. \underbrace{(\tilde{\alpha}_{G,1,i,l} = 1 \vee \tilde{\alpha}_{G,4,i,l} = 1)}_{=EP_{2,2}} \wedge \underbrace{(\tilde{\alpha}_{G,2,j,l} = 1 \vee \tilde{\alpha}_{G,5,j,l} = 1)}_{=EP_{2,3}} \right) \end{aligned} \quad (3.17b)$$

$$\begin{aligned} E_3 : \quad & \text{if “true”} \longrightarrow \text{directed edge } \{j, i\} \in \mathcal{E}_G \\ & \left(\underbrace{\tilde{\alpha}_{G,2,i,j} = 1}_{=EP_{3,1}} \right) \wedge \left(\nexists \{l\} \in \mathcal{G} \setminus \{i, j\} : \underbrace{\tilde{\alpha}_{G,2,i,l} = 1 \wedge \tilde{\alpha}_{G,1,j,l} = 1}_{=EP_{3,2}} \right) \end{aligned} \quad (3.17c)$$

$$\begin{aligned} E_4 : \quad & \text{if “true”} \longrightarrow \text{directed edge } \{j, i\} \in \mathcal{E}_G \\ & \left(\underbrace{\tilde{\alpha}_{G,5,i,j} = 1}_{=EP_{4,1}} \right) \wedge \left(\nexists \{l\} \in \mathcal{G} \setminus \{i, j\} : \right. \\ & \left. \underbrace{(\tilde{\alpha}_{G,2,i,l} = 1 \vee \tilde{\alpha}_{G,5,i,l} = 1)}_{=EP_{4,2}} \wedge \underbrace{(\tilde{\alpha}_{G,1,j,l} = 1 \vee \tilde{\alpha}_{G,4,j,l} = 1)}_{=EP_{4,3}} \right) \end{aligned} \quad (3.17d)$$

Table 3.1. Proposed sparse-topology recovery procedure composed of ordering conditions E_1 to E_4

According to Eq. (3.17b) of condition E_2 in Table 3.1, there is an edge directed from gene i to gene j in the GENIE estimate \mathcal{D}_G of the true DAG \mathcal{D} , given that the condition described by Eq. (3.17b) is true. In particular, assume that the gene pair $\{i, j\} \in \mathcal{G} : j > i$ has been classified to class $k = 4$ by the solution of the GENIE-problem, i.e., $\tilde{\alpha}_{G,4,i,j} = 1$. Furthermore, assume that there is no gene $l \in \mathcal{G} \setminus \{i, j\}$ for which, neither $\tilde{\alpha}_{G,1,i,l} = 1$ or $\tilde{\alpha}_{G,4,i,l} = 1$, nor $\tilde{\alpha}_{G,2,j,l} = 1$ or $\tilde{\alpha}_{G,5,j,l} = 1$ holds. Then parts $EP_{2,1}$, $EP_{2,2}$ and $EP_{2,3}$ of Eq. (3.17b) are true which yields that condition E_2 is true. In this case, conditions E_1 as well as conditions E_3 to E_4 are automatically false due to the multiple-choice constraint in Eq. (3.5c) of the GENIE-problem. Note that parts $EP_{2,2}$ and $EP_{2,3}$ being false reflect an ordering constellation that directly contradicts the existence of an edge from gene i to gene j , i.e., $\{i, j\}$. To see this, the following

argument is presented:

- $\tilde{\alpha}_{G,4,i,j} = 1$ implies that there is a subset of paths $\tilde{\mathcal{P}}_i$ of the set of all paths \mathcal{P}_i from i to R , i.e., $\tilde{\mathcal{P}}_i \subset \mathcal{P}_i$, such that any path in $\tilde{\mathcal{P}}_i$ traverses gene j .
- given that parts EP_{2,2} and EP_{2,3} are false, i.e., there is a gene l such that $\tilde{\alpha}_{G,1,i,l} = 1 \vee \tilde{\alpha}_{G,4,i,l} = 1$ is true and $\tilde{\alpha}_{G,2,j,l} = 1 \vee \tilde{\alpha}_{G,5,j,l} = 1$ is true, then this specific gene l is also traversed by the paths collected in set $\tilde{\mathcal{P}}_i$
- $\tilde{\alpha}_{G,2,j,l} = 1$ or $\tilde{\alpha}_{G,5,j,l} = 1$ being true implies that gene l is traversed by any path of $\tilde{\mathcal{P}}_i$ before gene j is traversed.

Consequently, genes i and j cannot be neighbours in DAG \mathcal{D}_G , i.e., there cannot be a directed edge $\{i, j\}$ in DAG \mathcal{D}_G linking genes i and j , since there is at least one gene l that separates genes i and j in DAG \mathcal{D}_G .

As stated by Eq. (3.17c) of condition E₃ in Table 3.1, there is an edge directed from gene j to gene i in the GENIE estimate \mathcal{D}_G of the true DAG \mathcal{D} , given that the condition described by Eq. (3.17c) is true. In particular, assume that the gene pair $\{i, j\} \in \mathcal{G} : j > i$ has been classified to class $k = 2$ by the solution of the GENIE-problem, i.e., $\tilde{\alpha}_{G,2,i,j} = 1$. Furthermore, assume that there is no gene $l \in \mathcal{G} \setminus \{i, j\}$ for which neither $\tilde{\alpha}_{G,2,i,l} = 1$ nor $\tilde{\alpha}_{G,1,j,l} = 1$ is true. Then parts EP_{3,1} and EP_{3,2} of Eq. (3.17c) are true which yields that condition E₃ is true. In this case, conditions E₁, E₂ as well as condition E₄ are automatically false due to the multiple-choice constraint in Eq. (3.5c) of the GENIE-problem. Note that part EP_{3,2} being false reflects an ordering constellation that directly contradicts the existence of an edge from gene j to gene i , i.e., $\{j, i\}$. In order to see this, an argument similar to the one used to reason the functionality of part EP_{1,2} in Eq. (3.17a) is presented in the following:

- $\tilde{\alpha}_{G,2,i,j} = 1$ implies that every path $P_\kappa^{(j)}$ from gene j to the reporter node R traverses gene i
- $\tilde{\alpha}_{G,2,i,l} = 1$ implies that every path $P_\zeta^{(l)}$ from gene l to the reporter node R traverses gene i as well
- $\tilde{\alpha}_{G,1,j,l} = 1$ implies that every path $P_\kappa^{(j)}$ from gene j to the reporter node R traverses gene l . Hence, any path from gene j to gene i traverses gene l .

In essence, genes i and j cannot be neighbours in DAG \mathcal{D}_G , i.e., there cannot be a directed edge $\{j, i\}$ in DAG \mathcal{D}_G linking genes i and j , since there is at least one gene l that separates genes i and j in DAG \mathcal{D}_G .

According to Eq. (3.17d) of condition E_4 in Table 3.1, there is an edge directed from gene j to gene i in the GENIE estimate \mathcal{D}_G of the true DAG \mathcal{D} , given that the condition depicted by Eq. (3.17d) is true. In particular, assume that the gene pair $\{i, j\} \in \mathcal{G} : j > i$ has been classified to class $k = 5$ by the solution of the GENIE-problem, i.e., $\tilde{\alpha}_{G,5,i,j} = 1$. Furthermore, assume that there is no gene $l \in \mathcal{G} \setminus \{i, j\}$ for which, neither $\tilde{\alpha}_{G,2,i,l} = 1$ or $\tilde{\alpha}_{G,5,i,l} = 1$, nor $\tilde{\alpha}_{G,1,j,l} = 1$ or $\tilde{\alpha}_{G,4,j,l} = 1$ holds. Then parts $EP_{4,1}$, $EP_{4,2}$ and $EP_{4,3}$ of Eq. (3.17d) are true which yields that condition E_4 is true. In this case, conditions E_1 to E_3 are automatically false due to the multiple-choice constraint in Eq. (3.5c) of the GENIE-problem. Note again that parts $EP_{4,2}$ and $EP_{4,3}$ being false reflect an ordering constellation directly contradicting the existence of an edge from gene j to gene i , i.e., $\{j, i\}$. To see this, a similar argument to the one used to reason the functionality of parts $EP_{2,2}$ and $EP_{2,3}$ in Eq. (3.17b) is presented in the following:

- $\tilde{\alpha}_{G,5,i,j} = 1$ implies that there is a subset of paths $\tilde{\mathcal{P}}_j$ of the set of all paths \mathcal{P}_j from j to R , i.e., $\tilde{\mathcal{P}}_j \subset \mathcal{P}_j$, such that any path in $\tilde{\mathcal{P}}_j$ traverses gene i .
- given that parts $EP_{4,2}$ and $EP_{4,3}$ are false, i.e., there is a gene l such that $\tilde{\alpha}_{G,2,i,l} = 1 \vee \tilde{\alpha}_{G,5,i,l} = 1$ is true and $\tilde{\alpha}_{G,1,j,l} = 1 \vee \tilde{\alpha}_{G,4,j,l} = 1$ is true, then this specific gene l is also traversed by the paths collected in set $\tilde{\mathcal{P}}_j$
- $\tilde{\alpha}_{G,1,j,l} = 1$ or $\tilde{\alpha}_{G,4,j,l} = 1$ being true implies that gene l is traversed by any path of $\tilde{\mathcal{P}}_j$ before gene i is traversed.

As a consequence, genes i and j cannot be neighbours in DAG \mathcal{D}_G , i.e., there cannot be a directed edge $\{j, i\}$ in DAG \mathcal{D}_G linking genes i and j , since there is at least one gene l that separates genes i and j in DAG \mathcal{D}_G .

It is important to remark that, given that $\tilde{\alpha}_{G,3,i,j} = 1$ is true, there cannot be an edge between genes i and j in DAG \mathcal{D}_G , since by definition this would pose a direct contradiction to the implications of $\tilde{\alpha}_{G,3,i,j} = 1$ being true.

For the purpose of reconstructing the edges that link a gene in DAG \mathcal{D}_G with the reporter level R , another ordering procedure is proposed in the following which also sticks to the paradigm of reconstructing the sparsest DAG of its DAG-family \mathcal{F}_{Ω_G} . Let \mathcal{M}_i be defined according to Eq. (3.18)

$$\mathcal{M}_i = \{l \mid l \in \mathcal{G} \setminus \{i\}, \tilde{\alpha}_{G,4,i,l} = 1\} \quad \text{for } i \in \mathcal{G} \quad (3.18)$$

as the set of all genes l out of the set of genes under study that are in class $k = 4$ with a specific gene i . Furthermore, let \mathcal{M}'_i as defined in Eq. (3.19) denote the set of all genes $l \in \mathcal{M}_i$ which are independent of at least one other gene $\tilde{l} \in \mathcal{M}_i$.

$$\mathcal{M}'_i = \left\{ l \mid l \in \mathcal{M}_i, \exists \tilde{l} \in \mathcal{M}_i \setminus \{l\} : \tilde{\alpha}_{G,3,l,\tilde{l}} = 1 \right\} \quad \text{for } i \in \mathcal{G} \quad (3.19)$$

In order to identify if there is an edge $\{i, R\} \in \mathcal{E}_{\mathcal{D},G}$ linking gene i with the reporter node R in DAG \mathcal{D}_G , condition E_R in Table 3.2, which is based on \mathcal{M}_i and \mathcal{M}'_i , has to be evaluated. Given that parts ER_1 and ER_2 of Eq. (3.20) in Table 3.2 are true for

$$E_R : \underbrace{\left(\nexists \{l\} \in \mathcal{G} \setminus \{i\} : \tilde{\alpha}_{G,1,i,l} = 1 \right)}_{=ER_1} \wedge \underbrace{\left(\mathcal{M}_i = \emptyset \vee \mathcal{M}'_i = \emptyset \right)}_{=ER_2} \quad \text{for } i \in \mathcal{G} \quad (3.20)$$

Table 3.2. Proposed “sparse” reporter node edge detection procedure characterized by condition E_R

gene $i \in \mathcal{G}$, yields that condition E_R is true for that specific gene. Hence, there is an edge $\{i, R\} \in \mathcal{E}_{\mathcal{D},G}$ from gene i to the reporter node R in the “GENIE reconstruction” \mathcal{D}_G of the true DAG \mathcal{D} . On the contrary, assuming that neither ER_1 nor ER_2 are true yields that condition E_R is false. Thus, there is no edge $\{i, R\} \in \mathcal{E}_{\mathcal{D},G}$ in DAG \mathcal{D}_G . In the following considerations, the conceptual idea behind the ordering conditions ER_1 and ER_2 as stated in Eq. (3.20) is explained:

- ER_1 is true for a specific gene $i \in \mathcal{G}$ given that there is no other gene l which is in a “linear pathway type” with gene i “upstream”, i.e., there is no gene l such that any path $P_\tau^{(i)} \in \mathcal{P}_i$ from gene i to R traverses gene l . In other words, given a specific $i \in \mathcal{G}$ ER_1 is true if there is no other gene l that can be arranged in a “linear pathway type” “downstream” of gene i , i.e., closer to the root node R
- ER_2 is true for a specific gene $i \in \mathcal{G}$ given that \mathcal{M}_i or \mathcal{M}'_i are empty. Note that given $\mathcal{M}_i = \emptyset$ it follows by definition that $\mathcal{M}'_i = \emptyset$ as well. The ordering condition stated by ER_2 reflects the paradigm to recover the sparsest DAG \mathcal{D}_G of the DAG-family \mathcal{F}_{Ω_G} specified by the GENIE-estimate Ω_G , i.e., A_G/\tilde{A}_G . To illustrate this, assume that $\mathcal{M}_i = \emptyset$, then $\mathcal{M}'_i = \emptyset$ and ER_2 is true. ER_1 being true together with $\mathcal{M}_i = \emptyset$, i.e., ER_2 being true, basically state that there is no other gene l that can be arranged “closer” to the root node R . In the case of $\mathcal{M}_i \neq \emptyset$, i.e., there are some genes that are hierarchically “downstream” of gene i

(in class $k = 4$), and ER_1 being true, testing whether $\mathcal{M}'_i = \emptyset$ evaluates if there are multiple “linear pathway types” hierarchically “downstream” of gene i that terminate in the root node R . If there are such multiple “linear pathway types”, then an edge from i to R is redundant in a sense that it is not necessary for DAG \mathcal{D}_G to represent Ω_G , i.e., A_G/\tilde{A}_G .

For the purpose of clarity, the topology recovery policy depicted by conditions E_1 to E_4 in Table 3.1 and condition E_R in Table 3.2 is explained in an intuitive manner in Examples 3.1.2 and 3.1.3.

Example 3.1.2. *Given DAG \mathcal{D}_3 and its representation in the domain of class indicator coefficients denoted as $\Omega_3 = \Omega(\mathcal{D}_3)$ which is given by:*

$$\Omega_3 = \left\{ \begin{aligned} &\omega_{1,2}^{(3)} = 4, \omega_{1,3}^{(3)} = 4, \omega_{1,4}^{(3)} = 4, \omega_{1,5}^{(3)} = 1; \\ &\omega_{2,3}^{(3)} = 1, \omega_{2,4}^{(3)} = 1, \omega_{2,5}^{(3)} = 1 \\ &\omega_{3,4}^{(3)} = 1, \omega_{3,5}^{(3)} = 1 \\ &\omega_{4,5}^{(3)} = 1 \end{aligned} \right\}$$

Furthermore, let $\mathcal{F}_{\Omega_3} = \{\mathcal{D}_{3a}, \mathcal{D}_{3b}, \mathcal{D}_{3c}, \mathcal{D}_{3d}\}$ denote the DAG-family characterized by Ω_3 which is displayed in Figure 3.4. The set of redundantly expanded class selection variables $\tilde{A}_{\mathcal{D}_3}$ corresponding to Ω_3 is given by:

$$\tilde{A}_{\mathcal{D}_3} = \left\{ \begin{aligned} &\tilde{\alpha}_{4,1,2} = 1, \dots, \tilde{\alpha}_{4,1,3} = 1, \dots, \tilde{\alpha}_{4,1,4} = 1, \dots, \tilde{\alpha}_{1,1,5} = 1, \dots, \\ &\tilde{\alpha}_{5,2,1} = 1, \dots, \tilde{\alpha}_{1,2,3} = 1, \dots, \tilde{\alpha}_{1,2,4} = 1, \dots, \tilde{\alpha}_{1,2,5} = 1, \dots, \\ &\tilde{\alpha}_{5,3,1} = 1, \dots, \tilde{\alpha}_{2,3,2} = 1, \dots, \tilde{\alpha}_{1,3,4} = 1, \dots, \tilde{\alpha}_{1,3,5} = 1, \dots, \\ &\tilde{\alpha}_{5,4,1} = 1, \dots, \tilde{\alpha}_{2,4,2} = 1, \dots, \tilde{\alpha}_{2,4,3} = 1, \dots, \tilde{\alpha}_{1,4,5} = 1, \dots, \\ &\tilde{\alpha}_{2,5,1} = 1, \dots, \tilde{\alpha}_{2,5,2} = 1, \dots, \tilde{\alpha}_{2,5,3} = 1, \dots, \tilde{\alpha}_{2,5,4} = 1, \dots, \end{aligned} \right\}$$

where it is important to remark that in the interest of readability only the non-zero class selection variables of $\tilde{A}_{\mathcal{D}_3}$ are displayed. Applying the ordering conditions E_1 to E_4 as well as E_R to $\tilde{A}_{\mathcal{D}_3}$ yields the sparsest DAG of the DAG-family \mathcal{F}_{Ω_3} which is given by DAG \mathcal{D}_{3a} of Figure 3.4. It is an easy task to show that the solid edges of the DAGs displayed in Figure 3.4 are recovered by the proposed edge detection policy as stated in Tables 3.1 and 3.2, respectively. However, edges e_1 and e_2 are not recovered by the proposed detection procedure as shown in the following:

- since genes g_1 and g_3 are in class $k = 4$, i.e., $\tilde{\alpha}_{4,1,3} = 1$, condition E_2 has to be evaluated. For gene g_2 , $\tilde{\alpha}_{4,1,2} = 1 \wedge \tilde{\alpha}_{2,3,2} = 1$ is true which violates condition E_2 . Hence, edge e_1 is not recovered.

- since genes g_1 and g_4 are in class $k = 4$, i.e., $\tilde{\alpha}_{4,1,4} = 1$, condition E_2 has to be evaluated again. For gene g_2 , $\tilde{\alpha}_{4,1,2} = 1 \wedge \tilde{\alpha}_{2,4,2} = 1$ is true which violates condition E_2 . Furthermore, for gene g_3 , $\tilde{\alpha}_{4,1,3} = 1 \wedge \tilde{\alpha}_{2,4,3} = 1$ is true which also violates condition E_2 . Thus, edge e_2 is not recovered.

In essence, the proposed edge detection procedure recovers DAG \mathcal{D}_{3a} based on $\tilde{A}_{\mathcal{D}_3}$, which is the sparsest DAG of its DAG-family \mathcal{F}_{Ω_3} .

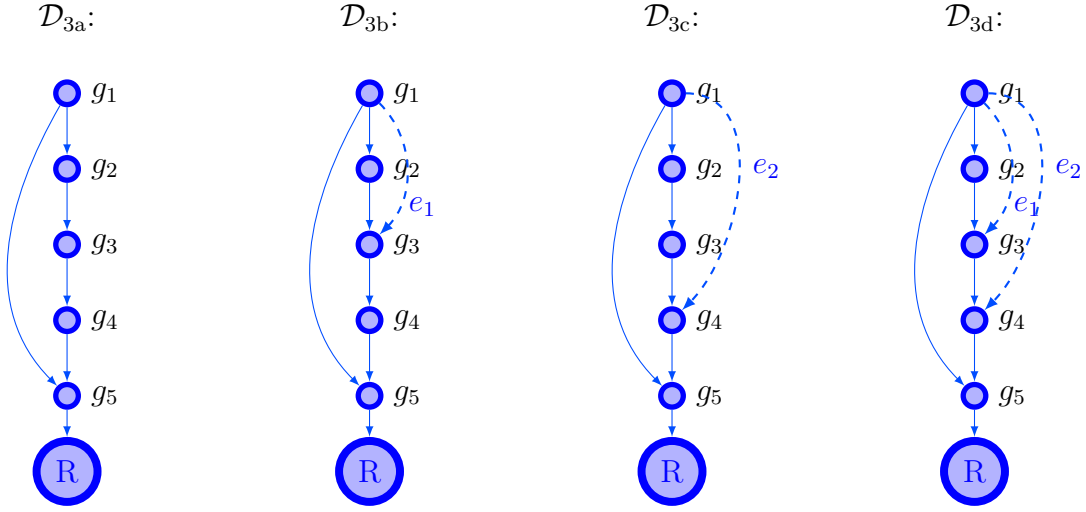


Figure 3.4. DAG-family \mathcal{F}_{Ω_3} composed of DAGs \mathcal{D}_{3a} , \mathcal{D}_{3b} , \mathcal{D}_{3c} and \mathcal{D}_{3d} and characterized by Ω_3 .

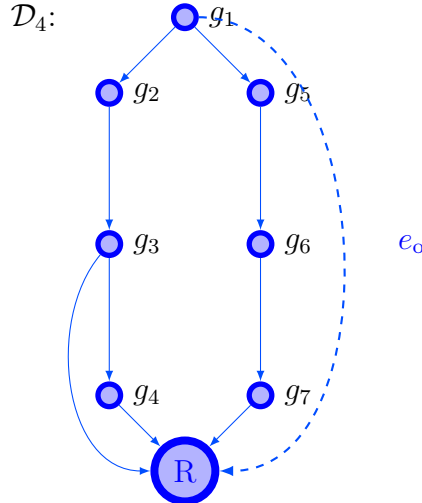


Figure 3.5. Example DAG \mathcal{D}_4 with redundant edge e_o to illustrate the sparsity promoting functionality of condition E_R depicted in Table 3.2.

Example 3.1.3. Given DAG-family \mathcal{F}_{Ω_4} characterized by Ω_4 which translates to the

redundantly expanded set of class selection variables \tilde{A}_4 described by

$$\begin{aligned} \tilde{A}_4 = \{ & \tilde{\alpha}_{4,1,2} = 1, \dots, \tilde{\alpha}_{4,1,3} = 1, \dots, \tilde{\alpha}_{4,1,4} = 1, \dots, \tilde{\alpha}_{4,1,5} = 1, \dots, \tilde{\alpha}_{4,1,6} = 1, \dots, \tilde{\alpha}_{4,1,7} = 1, \\ & \tilde{\alpha}_{5,2,1} = 1, \dots, \tilde{\alpha}_{1,2,3} = 1, \dots, \tilde{\alpha}_{4,2,4} = 1, \dots, \tilde{\alpha}_{3,2,5} = 1, \dots, \tilde{\alpha}_{3,2,6} = 1, \dots, \tilde{\alpha}_{3,2,7} = 1, \\ & \tilde{\alpha}_{5,3,1} = 1, \dots, \tilde{\alpha}_{2,3,2} = 1, \dots, \tilde{\alpha}_{4,3,4} = 1, \dots, \tilde{\alpha}_{3,3,5} = 1, \dots, \tilde{\alpha}_{3,3,6} = 1, \dots, \tilde{\alpha}_{3,3,7} = 1, \\ & \tilde{\alpha}_{5,4,1} = 1, \dots, \tilde{\alpha}_{5,4,2} = 1, \dots, \tilde{\alpha}_{5,4,3} = 1, \dots, \tilde{\alpha}_{3,4,5} = 1, \dots, \tilde{\alpha}_{3,4,6} = 1, \dots, \tilde{\alpha}_{3,4,7} = 1, \\ & \tilde{\alpha}_{5,5,1} = 1, \dots, \tilde{\alpha}_{3,5,2} = 1, \dots, \tilde{\alpha}_{3,5,3} = 1, \dots, \tilde{\alpha}_{3,5,4} = 1, \dots, \tilde{\alpha}_{1,5,6} = 1, \dots, \tilde{\alpha}_{1,5,7} = 1, \\ & \tilde{\alpha}_{5,6,1} = 1, \dots, \tilde{\alpha}_{3,6,2} = 1, \dots, \tilde{\alpha}_{3,6,3} = 1, \dots, \tilde{\alpha}_{3,6,4} = 1, \dots, \tilde{\alpha}_{2,6,5} = 1, \dots, \tilde{\alpha}_{1,6,7} = 1, \\ & \tilde{\alpha}_{5,7,1} = 1, \dots, \tilde{\alpha}_{3,7,2} = 1, \dots, \tilde{\alpha}_{3,7,3} = 1, \dots, \tilde{\alpha}_{3,7,4} = 1, \dots, \tilde{\alpha}_{2,7,5} = 1, \dots, \tilde{\alpha}_{2,7,6} = 1, \\ & \} \end{aligned}$$

where again only the non-zero elements of \tilde{A}_4 are displayed for the sake of readability. In order to reconstruct the sparsest DAG of the DAG-family \mathcal{F}_{Ω_4} conditions E_1 to E_4 of Table 3.1 have to be evaluated for each gene pair $\{i, j\} \in \mathcal{G} : j > i$ with $\mathcal{G} = \{1, \dots, 7\}$ to test for the existence of edges $\{i, j\}$ and $\{j, i\}$, respectively. Moreover, to identify all edges between some gene $i \in \mathcal{G}$ and the reporter node R condition E_R has to be evaluated for all genes i . In Figure 3.5, DAG \mathcal{D}_4 is displayed which is the sparsest DAG of DAG-family \mathcal{F}_{Ω_4} . Following the edge detection policy proposed in Tables 3.1 and 3.2, respectively, it can be easily shown that the set of solid edges in DAG \mathcal{D}_4 constitutes the sparsest DAG topology with respect to \mathcal{F}_{Ω_4} . The edges that link some gene with the reporter node R are given by $\{3, R\}$, $\{4, R\}$ and $\{7, R\}$. However, the dashed edge e_o connecting gene g_1 with the reporter node R is not detected, although it is valid with respect to \tilde{A}_4 . To see this, condition E_R is evaluated for gene g_1 :

- part ER_1 of condition E_R is obviously true
- $\mathcal{M}_1 = \{2, 3, 4, 5, 6, 7\}$ and $\mathcal{M}'_1 = \{2, 3, 4, 5, 6, 7\}$
- hence, ER_2 is false which finally yields that E_R is false as well
- edge e_o is not recovered

This results from the sparsity promoting part ER_2 of the ordering condition E_R in Table 3.2. Since gene g_1 is in class $k = 4$ to all other genes in DAG \mathcal{D}_4 , i.e., $\tilde{\alpha}_{4,1,l} = 1$ for all genes $l \in \mathcal{G} \setminus \{1\}$, it is hierarchically “upstream” of all other genes in \mathcal{D}_4 and there must always be at least one path from g_1 to R that is independent of gene l for all $l \in \mathcal{G} \setminus \{1\}$. Edge e_o yields a path from gene g_1 to the reporter node R which is independent of any other gene in DAG \mathcal{D}_4 . However, neglecting edge e_o , gene g_1 still has at least one independent path to R with respect to any other gene in the DAG. Consequently, edge e_o is redundant and cannot belong to the sparsest DAG out of the DAG-family \mathcal{F}_{Ω_4} .

Finally, the proposed graph topology reconstruction procedure, based on the GENIE-estimate A_G/\tilde{A}_G and the edge detection policy stated in Tables 3.1 and 3.2, is summarized as Algorithm 4.

Algorithm 4 Graph Topology Reconstruction

```

1: Initialize:
    $\tilde{\mathcal{E}}_{\mathcal{D},G} = \emptyset$ , set of genes under study  $\mathcal{G}$ ,  $\tilde{A}_G$ 
2: for  $\{i, j\} \in \mathcal{G} : j > i$  do
3:   if  $E_1$  “true” or  $E_2$  “true” then
4:     update GENIE-topology estimate:  $\tilde{\mathcal{E}}_{\mathcal{D},G} := \tilde{\mathcal{E}}_{\mathcal{D},G} \cup \{i, j\}$ 
5:   else if  $E_3$  “true” or  $E_4$  “true” then
6:     update GENIE-topology estimate:  $\tilde{\mathcal{E}}_{\mathcal{D},G} := \tilde{\mathcal{E}}_{\mathcal{D},G} \cup \{j, i\}$ 
7:   else
8:     no update of GENIE-topology estimate:  $\tilde{\mathcal{E}}_{\mathcal{D},G}$ 
   return  $\implies \tilde{\mathcal{E}}_{\mathcal{D},G}$ 
9: set:  $\mathcal{E}_{\mathcal{D},G} := \tilde{\mathcal{E}}_{\mathcal{D},G}$ 
10: for  $\{i\} \in \mathcal{G}$  do
11:   if  $E_R$  “true” then
12:      $\mathcal{E}_{\mathcal{D},G} := \mathcal{E}_{\mathcal{D},G} \cup \{i, R\}$ 
13: return  $\implies$  GENIE-topology estimate  $\mathcal{E}_{\mathcal{D},G}$ 

```

3.2 GI-GENIE Algorithm

In this section, the GI-GENIE algorithm, that learns the DAG/GI-network underlying the observed SK/DK data and the GI-profile data, respectively, is derived. In particular, the proposed GI-GENIE algorithm learns the hierarchical relationship class representation of the underlying GI-network/DAG, as well as the corresponding DAG-topology, jointly. Furthermore, by incorporating multiple data types such as SK/DK data as well as GI-profile data, the GI-GENIE algorithm yields better results than the GENIE algorithm in terms of estimation accuracy.

3.2.1 ILP-Formulation of the GI-GENIE Algorithm

Let the *edge selection variables* be defined as given below by Eq. (3.21)

$$\beta_{i,j} = \begin{cases} 1 & \exists \text{ edge between } i,j \\ 0 & \text{no edge} \end{cases} \quad \forall \{i,j\} \in \mathcal{G} : j > i \quad (3.21)$$

where $\beta_{i,j} = 1$ indicates that there is an undirected edge between genes i and j in DAG \mathcal{D} . Conversely, in the case of $\beta_{i,j} = 0$ there is no edge between genes i and j in DAG \mathcal{D} . Moreover, let the set of undirected edge selection variables be denoted by

$$B_{\mathcal{D}} = \bigcup_{\{i,j\} \in \mathcal{G} : j > i} \beta_{i,j}. \quad (3.22)$$

The set of edge selection variables $B_{\mathcal{D}}$ reflects the undirected skeleton of DAG \mathcal{D} , excluding all edges between some gene $i \in \mathcal{G}$ and the reporter level R . This is due to the fact that the edge selection variables $\beta_{i,j}$ are associated with the GI-profile data $\rho(i,j)$ which does not exist for the artificial root node in a DAG, i.e., the reporter level R . The problem of jointly learning the set of class selection variables $A_{\mathcal{D}}$ corresponding to DAG/GI-network \mathcal{D} which is associated with the minimal overall mismatch score, as well as the set of corresponding edge selection variables $B_{\mathcal{D}}$ that describes the undirected skeleton of \mathcal{D} , has been formulated in Eqs. (3.23) as an ILP and is denoted as

the GI-GENIE-problem as stated below

$$\begin{aligned} \min_{\{\alpha_{k,i,j}, \beta_{i,j}, z_{l,i,j}\}} \quad & \lambda_d \sum_{\{i,j\} \in \mathcal{G}: j > i} \sum_{k \in \mathcal{K}} s_k(i,j) \alpha_{k,i,j} - \lambda_c \sum_{\{i,j\} \in \mathcal{G}: j > i} \beta_{i,j} \rho(i,j) \\ & + \lambda_p \sum_{\{i,j\} \in \mathcal{G}: j > i} \beta_{i,j} \end{aligned} \quad (3.23a)$$

s. t.:

$$\alpha_{k,i,j} \in \{0, 1\} \quad \forall \{i,j\} \in \mathcal{G} : j > i, \quad \forall k \in \mathcal{K} \quad (3.23b)$$

$$\sum_{k \in \mathcal{K}} \alpha_{k,i,j} = 1, \quad \forall \{i,j\} \in \mathcal{G} : j > i \quad (3.23c)$$

$$\mathcal{L} \implies \text{class coupling constraints} \quad (3.23d)$$

$$\beta_{i,j} \in \{0, 1\} \quad \forall \{i,j\} \in \mathcal{G} : j > i \quad (3.23e)$$

$$z_{l,i,j} \in \{0, 1\} \quad \forall l \in \mathcal{G} \setminus \{i,j\}, \quad \forall \{i,j\} \in \mathcal{G} : j > i \quad (3.23f)$$

$$1 - \alpha_{3,i,j} \geq \beta_{i,j} \quad \forall \{i,j\} \in \mathcal{G} : j > i \quad (3.23g)$$

$$\mathcal{L}_c \implies \text{additional topology constraints} \quad (3.23h)$$

$$|\mathcal{G}| - 2 + \beta_{i,j} \geq 1 + \sum_{l \in \mathcal{G} \setminus \{i,j\}} z_{l,i,j} \quad \forall \{i,j\} \in \mathcal{G} : j > i \quad (3.23i)$$

where the solution to the GI-GENIE-problem as stated in Eqs. (3.23) is denoted by

$$A_{\text{GI}} = \bigcup_{\{i,j\} \in \mathcal{G}: j > i, k \in \mathcal{K}} \alpha_{\text{GI},k,i,j}, \quad B_{\text{GI}} = \bigcup_{\{i,j\} \in \mathcal{G}: j > i} \beta_{\text{GI},i,j} \quad (3.24)$$

and the parameters $\lambda_d, \lambda_c, \lambda_p$ are non-negative real weighting constants. The optimization objective of the GI-GENIE-problem is to learn the set of class selection variables A_{GI} which yields the lowest overall mismatch score $S_{\text{min,GI}}$ as defined in Eq. (3.25)

$$S_{\text{min,GI}} = \sum_{\{i,j\} \in \mathcal{G}: j > i} \sum_{k \in \mathcal{K}} \alpha_{\text{GI},k,i,j} s_k(i,j). \quad (3.25)$$

jointly with the corresponding set of edge selection variables B_{GI} that yields the lowest GI-profile mismatch pattern $P_{\text{min,GI}}$ as defined in Eq. (3.26)

$$P_{\text{min,GI}} = -\lambda_c \sum_{\{i,j\} \in \mathcal{G}: j > i} \beta_{\text{GI},i,j} \rho(i,j) + \lambda_p \sum_{\{i,j\} \in \mathcal{G}: j > i} \beta_{\text{GI},i,j}. \quad (3.26)$$

The parameters $\lambda_d, \lambda_c, \lambda_p$ balance the impact of the SK/DK data on the one hand and the GI-profile data $\rho(i, j)$ on the other hand on the estimation result, i.e., A_{GI} and B_{GI} . Furthermore, λ_d can be decomposed into two parts as $\lambda_d = \frac{1}{\max_{k,i,j}\{s_k(i,j)\}} \tilde{\lambda}_d$ and is used for dual purpose:

- weighting by $\frac{1}{\max_{k,i,j}\{s_k(i,j)\}}$ scales the magnitude of the SK/DK based mismatch scores $s_k(i, j)$ to the range $[0, 1]$ which is comparable to the magnitude of the GI-profile data $\rho(i, j) \in [-1, 1]$
- scaling by $\tilde{\lambda}_d$ trades off the impact of the SK/DK based mismatch scores $s_k(i, j)$ against the GI-profile data $\rho(i, j)$ on the estimation result

The parameters $\lambda_c, \lambda_p \in [0, 1]$, with $\lambda_c \geq \lambda_p$, are related with the GI-profile term

$$-\lambda_c \sum_{\{i,j\} \in \mathcal{G}: j > i} \beta_{i,j} \rho(i, j) + \lambda_p \sum_{\{i,j\} \in \mathcal{G}: j > i} \beta_{i,j}. \quad (3.27)$$

of the objective in Eq. (3.23) where the quotient $\frac{\lambda_p}{\lambda_c}$ defines the threshold of reward for setting the edge selection variables $\beta(i, j)$. In particular, given that $\rho(i, j) \geq \frac{\lambda_p}{\lambda_c}$ for a specific pair of genes $\{i, j\}$, setting $\beta(i, j) = 1$ for that pair is beneficial with respect to the optimization objective, since it decreases the GI-profile term. Equivalent to the GENIE-problem formulation in Eq. (3.5), constraints (3.23b) to (3.23d) of the GI-GENIE-problem ensure that the learned set of hierarchical relationship classes A_{GI} directly corresponds to a set of class indicator coefficients $\Omega_{GI} = \Omega(\mathcal{D}_{GI})$ that reflects DAG \mathcal{D}_{GI} which is an estimate of the true DAG/GI-network topology \mathcal{D} . The constraint in Eq. (3.23e) reflects the binary nature of the edge selection variables defined in Eq. (3.21). The binary variables $z_{l,i,j}$ as defined in constraint (3.23f) are auxiliary variables that are necessary for the joint estimation of A_{GI} and B_{GI} .

In general, the set of hierarchical class selection variables $A_{\mathcal{D}}$ and the corresponding set of edge selection variables $B_{\mathcal{D}}$ are highly coupled. This is the case, since $A_{\mathcal{D}}$ and $B_{\mathcal{D}}$ both describe the topology of DAG \mathcal{D} , however, in different representation domains.

In particular, given that $\beta_{i,j} = 0$ for some specific pair of genes $\{i, j\} \in \mathcal{G} : j > i$, then conditions E₁ to E₄ of Table 3.1 must be false for pair $\{i, j\}$ based on $\tilde{A}_{\mathcal{D}}$ which is the redundant expansion of $A_{\mathcal{D}}$ according to Eqs. (3.14)-(3.16). On the contrary, assuming that $\beta_{i,j} = 1$ for some specific pair of genes $\{i, j\} \in \mathcal{G} : j > i$, then the set $\tilde{A}_{\mathcal{D}}$ must be structured in such a way that exactly one condition of Table 3.1 is true for that specific pair $\{i, j\}$.

Conversely, assuming that $\tilde{A}_{\mathcal{D}}$ is structured in such a way that none condition of Table 3.1 is true for a specific pair of genes $\{i, j\}$, then the corresponding edge selection variable $\beta_{i,j}$ must be zero, i.e., $\beta_{i,j} = 0$. Moreover, given that the structure of $\tilde{A}_{\mathcal{D}}$ implies that there must be an edge between a specific pair of genes $\{i, j\}$, i.e., that exactly one condition depicted in Table 3.1 is true, then the corresponding edge selection variable $\beta_{i,j}$ must be one, i.e., $\beta_{i,j} = 1$.

Let the auxiliary parameters

$$q_{i,j} = \begin{cases} 1 & \rho(i, j) \geq \frac{\lambda_p}{\lambda_c} \\ 0 & \rho(i, j) < \frac{\lambda_p}{\lambda_c} \end{cases} \quad \forall \{i, j\} \in \mathcal{G} : j > i \quad (3.28)$$

indicate those pairs of genes $\{i, j\}$ whose GI-profile $\rho(i, j)$ is above the pre-defined threshold of reward $\frac{\lambda_p}{\lambda_c}$. The collection of auxiliary parameters $q_{i,j}$ as defined in Eq. (3.28) yields an estimate of the undirected skeleton of DAG \mathcal{D} based on GI-profile data only. In essence, constraints (3.23f) to (3.23i), together with the auxiliary parameters $q_{i,j}$, jointly model a minor modification of the edge detection policy proposed in Table 3.1 which is elucidated in the following.

Constraint (3.23g) reflects the most intuitive coupling rule among the class selection variables in $A_{\mathcal{D}}$ and the edge selection variables in $B_{\mathcal{D}}$. Given that $\alpha_{3,i,j} = 1$, i.e., i and j are independent, then the corresponding edge selection variable $\beta_{i,j}$ must be zero by definition, i.e., $\beta_{i,j} = 0$. On the contrary, assuming that $\beta_{i,j} = 1$, genes i and j cannot be independent. Hence, $\alpha_{3,i,j}$ must be zero. Note that under the assumptions that $\alpha_{3,i,j} = 1$ and $\beta_{i,j} = 0$, respectively, Eq. (3.23g) is always fulfilled and no implications are made by this constraint. In order to account for the above mentioned coupling between the class selection variables in $A_{\mathcal{D}}$ and the edge selection variables in $B_{\mathcal{D}}$, the constraints aggregated in \mathcal{L}_c model the edge detection conditions E₁ to E₄ of Table 3.1 which reflect the coupling among the selection variables $\alpha_{k,i,j}$ and $\beta_{i,j}$, respectively.

It is important to remark that the compact formulation of the proposed edge detection policy in Table 3.1 is based on the redundant expansion of class selection variables. An edge between genes i and j , for $i, j \in \mathcal{G} : j > i$, is recovered given that there is no other gene $l \in \mathcal{G}$ which violates conditions E₁ to E₄. This implies, that for any triplet of genes $\{i, j, l\} \in \mathcal{G}$ and its pairwise decomposition $\{i, j\}$, $\{i, l\}$ and $\{j, l\}$, the corresponding class selection variables $\alpha_{k,i,j}$, $\alpha_{k,i,l}$ and $\alpha_{k,j,l}$, for all $k \in \mathcal{K}$, exist which is true for the redundantly expanded set of class selection variables $\tilde{A}_{\mathcal{G}}$. In the case of reconstructing the DAG topology according to Algorithm 4, the redundant expansion of the set of class selection variables is not costly, since the proposed topology reconstruction procedure stated by Algorithm 4 and the estimation of the set of class selection variables $A_{\mathcal{G}}/\tilde{A}_{\mathcal{G}}$ by the GENIE-algorithm are sequential.

However, in the case of the GI-GENIE algorithm which jointly learns the hierarchical relationship class representation A_{GI} of DAG \mathcal{D} and the corresponding topology B_{GI} , it is computationally not reasonable to redundantly expand the class selection variable space. Thus, in the interest of computational tractability of the GI-GENIE problem, the edge detection conditions E_1 to E_4 of Table 3.1, that model the coupling among $A_{\mathcal{D}}$ and $B_{\mathcal{D}}$, have to be formulated for the cases $\{i, j, l\} \in \mathcal{G} : l > j > i$, $\{i, j, l\} \in \mathcal{G} : j > i > l$ and $\{i, j, l\} \in \mathcal{G} : j > l > i$.

In other words, the distinction into the above mentioned three cases accounts for the fact, that only for $\{i, j, l\} \in \mathcal{G} : l > j > i$ all class selection variables $\alpha_{k,i,j}$, $\alpha_{k,i,l}$ and $\alpha_{k,j,l}$, corresponding to the pairwise decomposition $\{i, j\}$, $\{i, l\}$ and $\{j, l\}$, exist. Hence, only for the case of $\{i, j, l\} \in \mathcal{G} : l > j > i$ the proposed edge recovery procedure in Table 3.1 models the coupling among $A_{\mathcal{D}}$ and $B_{\mathcal{D}}$ correctly. However, using the symmetry properties of the class selection variables described by Eq. (3.14) and Eq. (3.16), the edge detection conditions E_1 to E_4 can be straightforwardly translated to the cases $\{i, j, l\} \in \mathcal{G} : j > i > l$ and $\{i, j, l\} \in \mathcal{G} : j > l > i$, respectively. Hence, the coupling among $A_{\mathcal{D}}$ and $B_{\mathcal{D}}$ reflected by \mathcal{L}_c is completely modeled by coupling sets $\mathcal{L}_{c,1}$, $\mathcal{L}_{c,2}$ and $\mathcal{L}_{c,3}$ which correspond to the three gene-triplet decomposition cases. In essence, \mathcal{L}_c is given by

$$\mathcal{L}_c = \bigcup_{\kappa=1}^3 \{\mathcal{L}_{c,\kappa}\}. \quad (3.29)$$

In the following, the constraints for modeling the coupling among $A_{\mathcal{D}}$ and $B_{\mathcal{D}}$ for the case that $\{i, j, l\} \in \mathcal{G} : l > j > i$ are summarized by set $\mathcal{L}_{c,1}$ and explained in detail. The constraint sets $\mathcal{L}_{c,2}$ and $\mathcal{L}_{c,3}$, that model the cases $\{i, j, l\} \in \mathcal{G} : j > i > l$ and $\{i, j, l\} \in \mathcal{G} : j > l > i$, respectively, are stated in Appendix C.

Constraints (3.31a) and (3.31b) of the coupling rules set $\mathcal{L}_{c,1}$ model the condition E_1 of Table 3.1. For a pair of genes $\{i, j\} \in \mathcal{G} : j > i$ assume that $\beta_{i,j} = 0$ and $\alpha_{1,i,j} = 1$. Note that under those assumptions the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled, i.e., no restrictions are made by (3.31c)- (3.31k). Furthermore, the LHS of Eq. (3.31a) amounts to 1 which poses no restrictions on the RHS of Eq. (3.31a), i.e., $\alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} \leq 3$. However, in this case Eq. (3.23i) in the GI-GENIE problem enforces that at least one $z_{l,i,j}$ is not one, i.e., $z_{l,i,j} = 0$ for at least one $l \in \mathcal{G} \setminus \{i, j\}$. This forces the RHS of (3.31b) to amount to 1 for that particular gene l which in turn implies that the LHS of (3.31b) must be one as well, i.e., $\alpha_{1,i,l} + \alpha_{2,j,l} = 2$. Hence, $\alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} = 3$ which states a violation of condition E_1 that must occur in $A_{\mathcal{D}}$, since $\beta_{i,j} = 0$. On the other hand, assume that $\beta_{i,j} = 1$ and $\alpha_{1,i,j} = 1$. Again, note that under those assumptions the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled,

i.e., no restrictions are made by (3.31c)- (3.31k). In this case, $A_{\mathcal{D}}$ must be structured in such a way that there is no gene $l \in \mathcal{G} \setminus \{i, j\}$ such that condition E_1 is violated. Given that $\beta_{i,j} = 1$ and $\alpha_{1,i,j} = 1$, then the LHS of (3.31a) is zero which enforces the RHS of (3.31a) to be less than or equal to zero, i.e., $\alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} \leq 2$, which reflects the case of condition E_1 being not violated. In contrast to that, assuming that $\alpha_{1,i,j} = 1$ and $A_{\mathcal{D}}$ is structured in such a way that condition E_1 is false, then $\beta_{i,j}$ must be zero, i.e., $\beta_{i,j} = 0$. Hence, there must be at least one gene $l \in \mathcal{G} \setminus \{i, j\}$ for which $\alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} = 3$ which yields that the RHS of (3.31a) is 1. This implies that the LHS of (3.31a) must amount to 1 as well, i.e., $\beta_{i,j} = 0$. Furthermore, the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ as well as constraint (3.23i) are always fulfilled under the assumptions made. Finally, assuming that $\alpha_{1,i,j} = 1$ and $A_{\mathcal{D}}$ is structured in such a way that condition E_1 is true, then $\beta_{i,j}$ must be one, i.e., $\beta_{i,j} = 1$. Consequently, there cannot be genes $l \in \mathcal{G} \setminus \{i, j\}$ for which $\alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} = 3$ which yields that the RHS of (3.31a) is less than 1. Under the assumptions made, i.e., $\alpha_{1,i,j} = 1$ and $\alpha_{1,i,l} + \alpha_{2,j,l} < 2$, the LHS of (3.31b) is smaller or equal to $\frac{1}{2}$, while $\alpha_{1,i,j}$ is 1 at the RHS of (3.31b) as well. Therefore, to fulfill constraint (3.31b) $z_{l,i,j}$ must be 1 for all $l \in \mathcal{G} \setminus \{i, j\}$. This in turn yields that the RHS of (3.23i) amounts to $|\mathcal{G}| - 1$ forcing $\beta_{i,j} = 1$ in order to make the LHS of (3.23i) larger or equal than the RHS. Note again, that the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled under the assumptions made.

The constraints block (3.31c) to (3.31f) models condition E_2 of Table 3.1 along with a minor modification denoted as $E_{2,\text{mod}}$ and defined according to Eq. (3.30) for the case of $\{i, j, l\} \in \mathcal{G} : l > j > i$.

$$E_{2,\text{mod}} : \text{ if "true" } \longrightarrow \beta_{i,j} = 1; \text{ directed edge } \{i, j\} \in \mathcal{E}_G$$

$$\left(\alpha_{4,i,j} = 1 \right) \bigwedge \left(\nexists \{l\} \in \mathcal{G} \setminus \{i, j\} : (\alpha_{1,i,l} = 1) \bigwedge (\alpha_{2,j,l} = 1 \vee \alpha_{5,j,l} = 1) \right)$$
(3.30)

Constraints (3.31c) and (3.31d) reflect condition E_2 as stated in Table 3.1 while constraints (3.31e) and (3.31f) reflect the “relaxed” version of condition E_2 , i.e., $E_{2,\text{mod}}$, which is not restricted to estimate the sparsest DAG topology.

In particular, $E_{2,\text{mod}}$ allows the reconstruction of all edges that do not create a topology, contradicting $A_{\mathcal{D}}$, given that those edges decrease the GI-profile term in the GI-GENIE objective. For instance, edges e_1 and e_2 of DAGs \mathcal{D}_{3b} to \mathcal{D}_{3d} , respectively, as displayed in Figure 3.4, could be recovered given that the GI-profile term of the GI-GENIE objective, as displayed by Eq. (3.27), is decreased. The auxiliary parameters $q_{i,j}$ as defined in Eq. (3.28) control whether the constraints related to E_2 , i.e., (3.31c) and (3.31d), or the constraints related to $E_{2,\text{mod}}$, i.e., (3.31e) and (3.31f), are “active”. Particularly,

given that $q_{i,j} = 0$ constraints (3.31e) and (3.31f), i.e., $E_{2,\text{mod}}$, are always fulfilled irrespective of the choice of class/edge selection variables, while constraints (3.31c) and (3.31d), i.e., E_2 , model the coupling among the class/edge selection variables. On the contrary, given that $q_{i,j} = 1$ constraints (3.31e) and (3.31f), i.e., $E_{2,\text{mod}}$, reflect the coupling among the class/edge selection variables, while constraints (3.31c) and (3.31d), i.e., E_2 , are always fulfilled irrespective of the choice of class/edge selection variables.

$$\mathcal{L}_{c,1} = \left\{ \begin{array}{l} 1 - \beta_{i,j} \geq \alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} - 2 \\ \frac{1}{2}(\alpha_{1,i,l} + \alpha_{2,j,l}) \geq \alpha_{1,i,j} - z_{l,i,j} \end{array} \right. \quad (3.31a)$$

$$\frac{1}{2}(\alpha_{1,i,l} + \alpha_{2,j,l}) \geq \alpha_{1,i,j} - z_{l,i,j} \quad (3.31b)$$

$$1 - \beta_{i,j} + q_{i,j} \geq \alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} - 2 \quad (3.31c)$$

$$\frac{1}{2}(\alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l}) \geq \alpha_{4,i,j} - z_{l,i,j} - q_{i,j} \quad (3.31d)$$

$$2 - \beta_{i,j} - q_{i,j} \geq \alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} - 2 \quad (3.31e)$$

$$\frac{1}{2}(\alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l}) \geq \alpha_{4,i,j} - z_{l,i,j} - 1 + q_{i,j} \quad (3.31f)$$

$$1 - \beta_{i,j} \geq \alpha_{2,i,j} + \alpha_{2,i,l} + \alpha_{1,j,l} - 2 \quad (3.31g)$$

$$\frac{1}{2}(\alpha_{2,i,l} + \alpha_{1,j,l}) \geq \alpha_{2,i,j} - z_{l,i,j} \quad (3.31h)$$

$$1 - \beta_{i,j} + q_{i,j} \geq \alpha_{5,i,j} + \alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{1,j,l} + \alpha_{4,j,l} - 2 \quad (3.31i)$$

$$\frac{1}{2}(\alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{1,j,l} + \alpha_{4,j,l}) \geq \alpha_{5,i,j} - z_{l,i,j} - q_{i,j} \quad (3.31j)$$

$$2 - \beta_{i,j} - q_{i,j} \geq \alpha_{5,i,j} + \alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{1,j,l} - 2 \quad (3.31k)$$

$$\frac{1}{2}(\alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{1,j,l}) \geq \alpha_{5,i,j} - z_{l,i,j} - 1 + q_{i,j} \quad (3.31l)$$

$$\left. \right\} \forall \{i, j, l\} \in \mathcal{G} : l > j > i$$

For a pair of genes $\{i, j\} \in \mathcal{G} : j > i$ assume that $\beta_{i,j} = 0$, $\alpha_{4,i,j} = 1$ and $q_{i,j} = 0$. Note that under those assumptions the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled. Furthermore, the LHS of Eq. (3.31c) amounts to 1 which poses no restrictions on the RHS of Eq. (3.31c), i.e., $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} \leq 3$. However, in this case Eq. (3.23i) in the GI-GENIE problem enforces again that at least one $z_{l,i,j}$ is not one, i.e., $z_{l,i,j} = 0$, for at least one $l \in \mathcal{G} \setminus \{i, j\}$. This forces the RHS

of (3.31d) to amount to 1 for that particular gene l which in turn implies that the LHS of (3.31d) must be one as well, i.e., $\alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 2$. Thus, $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 3$ which states a violation of condition E_2 which must occur in $A_{\mathcal{D}}$, since $\beta_{i,j} = 0$. Contrary, assume that $\beta_{i,j} = 1$, $\alpha_{4,i,j} = 1$ and $q_{i,j} = 0$. Note that under those assumptions the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled. In this case, $A_{\mathcal{D}}$ must be structured in such a way that there is no gene $l \in \mathcal{G} \setminus \{i, j\}$ such that condition E_2 is violated. With $\beta_{i,j} = 1$, $\alpha_{4,i,j} = 1$ and $q_{i,j} = 0$, the LHS of (3.31c) is zero which yields that the RHS of (3.31c) must be less than or equal to zero, i.e., $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} \leq 2$, which reflects the case of condition E_2 being not violated. Conversely, assuming that $\alpha_{4,i,j} = 1$, $q_{i,j} = 0$ and $A_{\mathcal{D}}$ is structured in such a way that condition E_2 is false, then $\beta_{i,j}$ must be zero, i.e., $\beta_{i,j} = 0$. Hence, there must be at least one gene $l \in \mathcal{G} \setminus \{i, j\}$ for which $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 3$ which yields that the RHS of (3.31c) is 1. This implies that the LHS of (3.31c) must amount to 1 as well, i.e., $\beta_{i,j} = 0$. Moreover, the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ as well as constraint (3.23i) are always fulfilled under the assumptions made. Finally, assuming that $\alpha_{4,i,j} = 1$, $q_{i,j} = 0$ and $A_{\mathcal{D}}$ is structured in such a way that condition E_2 is true, then $\beta_{i,j}$ must be one, i.e., $\beta_{i,j} = 1$. Consequently, there cannot be genes $l \in \mathcal{G} \setminus \{i, j\}$ for which $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 3$. Hence, that causes the RHS of (3.31c) to be less than 1. With the assumptions that $\alpha_{4,i,j} = 1$ and $\alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} < 2$, the LHS of (3.31d) is smaller or equal to $\frac{1}{2}$, while $\alpha_{4,i,j}$ is 1 at the RHS of (3.31d). Therefore, to fulfill constraint (3.31d) $z_{l,i,j}$ must be 1 for all $l \in \mathcal{G} \setminus \{i, j\}$. This in turn results in the RHS of (3.23i) to amount to $|\mathcal{G} - 1|$ forcing $\beta_{i,j} = 1$ in order to make the LHS of (3.23i) larger or equal than the RHS. Note again, that the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled under the assumptions made.

Now, assuming that $\beta_{i,j} = 0$, $\alpha_{4,i,j} = 1$ and $q_{i,j} = 1$, then condition $E_{2,\text{mod}}$, given by constraints (3.31e), (3.31f), reflects the coupling among the class/edge selection variables. Again, under the assumptions made the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled. With $\beta_{i,j} = 0$ and $q_{i,j} = 1$, the LHS of Eq. (3.31e) amounts to 1 which poses no restrictions on the RHS of Eq. (3.31e), i.e., $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} \leq 3$. However, in this case Eq. (3.23i) in the GI-GENIE problem enforces again that at least one $z_{l,i,j}$ is not one, i.e., $z_{l,i,j} = 0$ for at least one $l \in \mathcal{G} \setminus \{i, j\}$. This forces the RHS of (3.31f) to amount to 1 for that particular gene l which in turn implies that the LHS of (3.31f) must be one as well, i.e., $\alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 2$. Thus, $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 3$ which yields a violation of condition $E_{2,\text{mod}}$ and no edge between genes i, j can exist in the reconstructed DAG. Conversely, assume that $\beta_{i,j} = 1$, $\alpha_{4,i,j} = 1$ and $q_{i,j} = 1$. Note that under those assumptions the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled. In this case, $A_{\mathcal{D}}$ must be structured in such a way that there is no gene $l \in \mathcal{G} \setminus \{i, j\}$ such that condition $E_{2,\text{mod}}$ is violated. With

$\beta_{i,j} = 1$, $\alpha_{4,i,j} = 1$ and $q_{i,j} = 1$, the LHS of (3.31e) is zero which yields that the RHS of (3.31e) must be less than or equal to zero, i.e., $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} \leq 2$, which reflects the case of condition $E_{2,\text{mod}}$ being not violated. Hence, $A_{\mathcal{D}}$ is structured in such a way that $E_{2,\text{mod}}$ is true. Otherwise, given that $\alpha_{4,i,j} = 1$, $q_{i,j} = 1$ and $A_{\mathcal{D}}$ is structured in such a way that condition $E_{2,\text{mod}}$ is false, then $\beta_{i,j}$ must be zero, i.e., $\beta_{i,j} = 0$. Hence, there must be at least one gene $l \in \mathcal{G} \setminus \{i, j\}$ for which $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 3$ which yields that the RHS of (3.31e) is 1. This implies that the LHS of (3.31e) must amount to 1 as well, i.e., $\beta_{i,j} = 0$. Moreover, the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ as well as constraint (3.23i) are always fulfilled under the assumptions made. Finally, assuming that $\alpha_{4,i,j} = 1$, $q_{i,j} = 1$ and $A_{\mathcal{D}}$ is structured in such a way that condition $E_{2,\text{mod}}$ is true, then $\beta_{i,j}$ must be one, i.e., $\beta_{i,j} = 1$. Consequently, there cannot be genes $l \in \mathcal{G} \setminus \{i, j\}$ for which $\alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} = 3$. Thus this causes the RHS of (3.31e) to be less than 1. With $\alpha_{4,i,j} = 1$ and $\alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} < 2$, the LHS of (3.31f) is smaller or equal to $\frac{1}{2}$, while $\alpha_{4,i,j}$ is 1 at the RHS of (3.31f). Therefore, for constraint (3.31f) being fulfilled, $z_{l,i,j}$ must be 1 for all $l \in \mathcal{G} \setminus \{i, j\}$. This in turn results in the RHS of (3.23i) to amount to $|\mathcal{G} - 1|$ forcing $\beta_{i,j} = 1$ in order to make the LHS of (3.23i) larger or equal than the RHS. Note again, that the remaining constraints in $\mathcal{L}_{c,1}/\mathcal{L}_c$ are always fulfilled under the assumptions made.

Similarly to condition $E_{2,\text{mod}}$, condition $E_{4,\text{mod}}$ as defined by Eq. (3.32) for the case of $\{i, j, l\} \in \mathcal{G} : l > j > i$ models a relaxation of condition E_4 of Table 3.1 which allows the reconstruction of all edges that do not create a topology, contradicting $A_{\mathcal{D}}$, given that those edges decrease the GI-profile term in the GI-GENIE objective.

$$E_{4,\text{mod}} : \quad \text{if "true"} \longrightarrow \beta_{i,j} = 1; \text{ directed edge } \{j, i\} \in \mathcal{E}_{\mathcal{G}} \\ \left(\alpha_{5,i,j} = 1 \right) \bigwedge \left(\nexists \{l\} \in \mathcal{G} \setminus \{i, j\} : (\alpha_{2,i,l} = 1 \vee \alpha_{5,i,l} = 1) \bigwedge (\alpha_{1,j,l} = 1) \right) \quad (3.32)$$

Constraints (3.31g) and (3.31h) model condition E_3 of Table 3.1 and can be explained with the same line of argument that was used to elucidate constraints (3.31a) and (3.31b), respectively. Furthermore, constraints (3.31i) and (3.31j) reflect condition E_4 while constraints (3.31k) and (3.31l) model condition $E_{4,\text{mod}}$. Both constraint blocks, i.e., (3.31i) together with (3.31j) and (3.31k) together with (3.31l), can be explained in the same fashion as constraints (3.31c), (3.31d) and (3.31e), (3.31f), respectively.

Based on the solution of the GI-GENIE-problem, i.e., A_{GI} and B_{GI} , the topology $\mathcal{E}_{\mathcal{D},\text{GI}}$ of the GI-GENIE-estimate \mathcal{D}_{GI} of the true DAG \mathcal{D} can be computed according to Algorithm 5.

Algorithm 5 GI-GENIE Topology Reconstruction

```

1: Initialize:
    $\tilde{\mathcal{E}}_{\text{GI}} = \emptyset, A_{\text{GI}}, B_{\text{GI}}, \mathcal{G}$ 
2: for  $\{i, j\} \in \mathcal{G} : j > i$  do
3:
4:   if  $\alpha_{\text{GI},1,i,j}\beta_{\text{GI},i,j} > 0$  or  $\alpha_{\text{GI},4,i,j}\beta_{\text{GI},i,j} > 0$  then
5:      $\tilde{\mathcal{E}}_{\text{D,GI}} := \tilde{\mathcal{E}}_{\text{D,GI}} \cup \{i, j\}$ 
6:
7:   else if  $\alpha_{\text{GI},2,i,j}\beta_{\text{GI},i,j} > 0$  or  $\alpha_{\text{GI},5,i,j}\beta_{\text{GI},i,j} > 0$  then
8:      $\tilde{\mathcal{E}}_{\text{D,GI}} := \tilde{\mathcal{E}}_{\text{D,GI}} \cup \{j, i\}$ 
9:
10:  else
11:    no update of GI-GENIE-topology estimate:  $\tilde{\mathcal{E}}_{\text{D,GI}} := \tilde{\mathcal{E}}_{\text{D,GI}} \cup \emptyset$ 
12:  return  $\implies \tilde{\mathcal{E}}_{\text{D,GI}}$ 
13: Set:  $\mathcal{E}_{\text{D,GI}} := \tilde{\mathcal{E}}_{\text{D,GI}}$ 
14: for  $\{i\} \in \mathcal{G}$  do
15:   if  $E_{\text{R}}$  "true" then
16:      $\mathcal{E}_{\text{D,GI}} := \mathcal{E}_{\text{D,GI}} \cup \{i, R\}$ 
17: return  $\implies$  GI-GENIE-topology estimate  $\mathcal{E}_{\text{D,GI}}$ 

```

3.3 Scalability Techniques

Due to the combinatorial nature of the GENIE problem in Eq. (3.5) as well as the GI-GENIE problem in Eq. (3.23), the proposed GENIE and GI-GENIE algorithms, respectively, cannot be applied to large-scale data sets, since the number of candidate solutions grows exponentially with the number of genes for both algorithms. However, the GENIE and the GI-GENIE algorithm, respectively, can be embedded in a "divide-and-conquer" approach that makes use of those proposed algorithms in an iterative fashion. In this section, a sequential scalability procedure called SEQSCA is introduced, that estimates the GI-network of simplified gene-gene interaction types for large sets of genes based on DAG topology estimates provided by the GENIE/GI-GENIE algorithm of small-size gene subsets. In particular, the SEQSCA method yields a GI-network topology estimate which is a fully connected graph of weighted edges. In this GI-network topology estimate, the empirical probability of two genes to interact is described by the weight of the edge that links the associated genes with each other. Hence, the SEQSCA algorithm does not yield GI-network estimates that follow the terms of [BJW⁺10] and the associated data model \mathfrak{M}_{B} . Ultimately, the SEQSCA method provides a tradeoff between interaction model complexity and scalability in the sense that only one interaction type can be distinguished for the benefit of the ability to process large sets of genes.

The large set of genes \mathcal{G} is decomposed into subsets \mathcal{G}_s of equal size G_S for all $s \in \{1, \dots, S\}$ where $G_S \ll |\mathcal{G}| = G$. The subsets \mathcal{G}_s are chosen according to the famous urn-model [KB97]. In particular, the sample space $\Pi_{\mathcal{G}_s}$ of the urn-model random process without replacement, which the subsets \mathcal{G}_s are drawn from, is defined according to Eq. (3.33)

$$\Pi_{\mathcal{G}_s} = \left\{ \underbrace{(g_{i_1}, \dots, g_{i_{G_S}})}_{=\pi_i} \mid i_1, \dots, i_{G_S} \in \{1, \dots, G\}, \text{ with } i_l \neq i_m \right. \\ \left. \text{for } l \neq m, \text{ and } l, m \in \{1, \dots, G_S\} \right\} \quad (3.33)$$

where π_i represents a particular element of the sample space that is a G_S -dimensional tuple of genes, i.e., a candidate subset \mathcal{G}_s . A specific subset \mathcal{G}_s is drawn from $\Pi_{\mathcal{G}_s}$ as π_i with probability

$$\mathcal{G}_s \sim p(\pi_i) = \frac{1}{|\Pi_{\mathcal{G}_s}|} = \frac{1}{\frac{G!}{(G-G_S)!}} = \frac{(G-G_S)!}{G!} \quad \forall \pi_i \in \Pi_{\mathcal{G}_s}, \quad s \in \{1, \dots, S\} \quad (3.34)$$

In plain words, each gene in \mathcal{G}_s is drawn from \mathcal{G} with equal probability and without replacement.

For each subset of genes \mathcal{G}_s the topology $\mathcal{E}_{\mathcal{D}_s}$ of the corresponding DAG/GI-network \mathcal{D}_s is estimated by the GENIE/GI-GENIE algorithm, respectively. Furthermore, based on $\mathcal{E}_{\mathcal{D}_s}$ the corresponding adjacency matrix $\mathbf{M}_s \in \{0, 1\}^{G_S \times G_S}$ of DAG \mathcal{D}_s is computed according to Eq. (3.35)

$$\mathbf{M}_s(i, j) = \begin{cases} 1 & \text{if } \{i, j\} \in \mathcal{E}_{\mathcal{D}_s} \text{ or } \{j, i\} \in \mathcal{E}_{\mathcal{D}_s} \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in \mathcal{G}_s, \quad s \in \{1, \dots, S\}. \quad (3.35)$$

Let $\mathbf{M} \in [0, 1]^{G \times G}$ denote the “final” reliability matrix of the large-scale set of genes \mathcal{G} , where each entry $\mathbf{M}(i, j)$ of the \mathbf{M} models the empirical probability that the corresponding pair of genes $\{i, j\} \in \mathcal{G}$ interacts with each other. It is important to remark that the reliability matrix \mathbf{M} reflects a GI-network estimate of the large-scale set of genes \mathcal{G} which is a fully connected graph where each undirected edge $\{i, j\} = \{j, i\}$ for all pairs of genes $\{i, j\} \in \mathcal{G}$ is weighted with the empirical probability of interaction for those specific pairs $\{i, j\} \in \mathcal{G}$, i.e., weighted with the corresponding $\mathbf{M}(i, j)$. Hence, the GI-network described by \mathbf{M} is not a DAG.

The reliability matrix \mathbf{M} is computed in an iterative fashion based on the sequence of adjacency matrices \mathbf{M}_s corresponding to gene subsets \mathcal{G}_s , respectively, for $s = 1, \dots, S$. In particular, the “initial” unnormalized reliability matrix $\mathbf{M}^{(0)}$ is sequentially updated according to Eq. (3.36)

$$\mathbf{M}^{(s+1)}(i, j) = \mathbf{M}^{(s)}(i, j) + \mathbf{M}_s(\kappa_i, \kappa_j) \quad \forall i, j \in \mathcal{G}_s \quad (3.36)$$

where $\kappa_i \in \{1, \dots, G_S\}$ for all $i \in \mathcal{G}_s$ and $\cup_{\kappa_i} = \{1, \dots, G_S\}$. Essentially, the unnormalized reliability matrix $\mathbf{M}^{(s)}$ at iteration s is updated at each entry $i, j \in \mathcal{G}_s$ by incrementing $\mathbf{M}^{(s)}(i, j)$ by the corresponding adjacency matrix entry ($\mathbf{M}_s(\kappa_i, \kappa_j)$). Finally, unnormalized reliability matrix $\mathbf{M}^{(S)}$ at iteration S is normalized to yield the (normalized) reliability matrix \mathbf{M} according to Eq. (3.37)

$$\mathbf{M}(i, j) = \mathbf{M}^{(S)}(i, j) / \max\{1, c_{i,j}^{(S)}\} \quad \forall \{i, j\} \in \mathcal{G} \quad (3.37)$$

where $c_{i,j}^{(S)}$ denotes the number of how frequent the two genes i, j of set \mathcal{G} jointly occurred in the sampled subsets $\mathcal{G}_0, \dots, \mathcal{G}_S$. The entire procedure described in this section is denoted as the sequential scalability (SEQSCA) algorithm and summarized in Algorithm 6.

Algorithm 6 Sequential Scalability Technique – SEQSCA

```

1: Initialize:
    $\mathbf{M}^{(0)} = \mathbf{0}_{G \times G}$ ,  $\mathbf{M}_{s=0} = \mathbf{0}_{G_S \times G_S}$ ; frequency counter  $c_{i,j}^{(0)} = 0$ ,  $s = 0$ 
2: for  $s \leq S$  do
3:   select  $\mathcal{G}_s$  of size  $G_S$  according to Eq. (3.34)
4:   estimate  $\mathcal{E}_{\mathcal{D}_s}$  of set  $\mathcal{G}_s$  (e.g. with GENIE/GI-GENIE)
5:   translate  $\mathcal{E}_{\mathcal{D}_s}$  to  $\mathbf{M}_s$  according to Eq. (3.35)
6:   update reliability matrix  $\mathbf{M}^{(s)}$  according to Eq. (3.36)
7:   update:  $c_{i,j}^{(s+1)} \leftarrow c_{i,j}^{(s)} + 1$  for all  $\{i, j\} \in \mathcal{G}_s$ 
8:   update:  $s \leftarrow s + 1$ 
   return  $\Rightarrow \mathbf{M}^{(S)}$ 
9: normalize  $\mathbf{M}^{(S)}$  to magnitude space  $[0, 1]$ :
    $\mathbf{M}(i, j) = \mathbf{M}^{(S)}(i, j) / \max\{1, c_{i,j}^{(S)}\} \quad \forall \{i, j\} \in \mathcal{G}$ 
10: return  $\mathbf{M}$ 

```

3.4 Simulation Results

In this section, simulation results are presented in order to evaluate the performance of the proposed ILP-algorithms (GENIE/GI-GENIE) described in Section 3.1 and Section 3.2, respectively. In particular, in Subsection 3.4.1 the proposed ILP-algorithms are evaluated in terms of synthetic data and compared to the method of [BJW⁺10] that is based on the AIS algorithm presented in Section 2.3. In Subsection 3.4.2, the proposed ILP-algorithms as well as the AIS based method of [BJW⁺10] are applied to large-scale real data along with the SEQSCA-method of Section 3.3 to compute a “simplified” GI-network according to Section 3.3.

3.4.1 Synthetic Data Results

In order to assess the graph learning performance of the proposed ILPs and the benchmark method of [BJW⁺10], Monte Carlo (MC) simulations with respect to the signal-to-noise-ratio (SNR) have been conducted on synthetic data. As measures to evaluate the quality of the network topology estimates $\hat{\mathcal{E}}_{\mathcal{D}}$, the performance metrics P_{ed} depicted in Eq. (3.38) and P_{mis} depicted in Eq. (3.39) are used. In particular, P_{ed} describes the number of learned edges in $\hat{\mathcal{E}}_{\mathcal{D}}$ that do not exist in the true topology $\mathcal{E}_{\mathcal{D}}$ normalized by the number of edges in the true topology. Hence, P_{ed} describes the normalized number of erroneously learned edges. Contrarily, P_{mis} describes the number of missing edges in $\hat{\mathcal{E}}_{\mathcal{D}}$ normalized by the number of edges in the true topology.

$$P_{\text{ed}} = \frac{\left| \left(\mathcal{E}_{\mathcal{D}} \cup \hat{\mathcal{E}}_{\mathcal{D}} \right) \setminus \mathcal{E}_{\mathcal{D}} \right|}{|\mathcal{E}_{\mathcal{D}}|} \quad (3.38)$$

$$P_{\text{mis}} = \frac{\left| \left(\mathcal{E}_{\mathcal{D}} \cup \hat{\mathcal{E}}_{\mathcal{D}} \right) \setminus \hat{\mathcal{E}}_{\mathcal{D}} \right|}{|\mathcal{E}_{\mathcal{D}}|} \quad (3.39)$$

Note that the DAG topology estimate \mathcal{E}_{G} provided by the GENIE algorithm is computed according to Algorithm 4, whereas the GI-GENIE estimate of the DAG topology, i.e., \mathcal{E}_{GI} , is computed according to Algorithm 5. Furthermore, the AIS topology estimate is denoted as \mathcal{E}_{AIS} .

In systems biology, the most frequently considered cell function for conducting SK/DK experiments on bacteria is *bacteria colony growth/colony growth* that serves as a proxy of *bacteria fitness*. The *colony growth* is measured as the area of the fixed-size Petri dish that is covered by the bacteria colony. Hence, the observed *colony growth*, i.e., the covered area, varies between s_{min} and a maximum area s_{max} which is dictated by the size of the Petri dish. Hence, any observed SK/DK data is related to a covered Petri dish area which is situated between s_{min} and s_{max} . Given \mathcal{G} , that is the set of genes under study, the following steps have to be conducted to generate first a DAG/GI-network according to the topology constraints of [BJW⁺10] and secondly generate synthetic data for the MC simulations that stems from this DAG/GI-network:

Step 1:

Generate a DAG $\mathcal{D} = \left(\mathcal{G} \cup \{R\}, \mathcal{E}_{\mathcal{D}} \right)$ according to the topology constraints of [BJW⁺10] as described in Section 2.2. On a high level, the DAG generation

procedure according to [BJW⁺10] is composed of two stages: (I) generate a minimal DAG that is connected in the sense that any leaf node only has one outgoing edge and no incoming edge, and further, any other node, excluding the root node R , has exactly one outgoing edge and one incoming edge (II) apply a densification stage where the minimal DAG is densified by adding potentially multiple edges to the previously generated minimal DAG.

Step 2:

Given the generated DAG topology $\mathcal{E}_{\mathcal{D}}$, the hierarchical relationship classes $\alpha_{k,i,j}$ are identified according to conditions C_1 to C_5 of Table 2.1 for all gene pairs $\{i, j\} \in \mathcal{G} : j > i$

Step 3:

To generate the SK data $R(i) \forall i \in \mathcal{G}$, first the noise-free SK denoted by $\tilde{R}(i)$ is generated and distorted by Gaussian noise afterwards. Particularly, the noise-free SK $\tilde{R}(i)$ is drawn from a uniform distribution over the interval $[s_{\min}, s_{\max}]$, i.e., $\tilde{R}(i) \sim \mathcal{U}(s_{\min}, s_{\max})$ and is distorted afterwards by a Gaussian noise process denoted as $n_{\text{SK}}(i) \sim \mathcal{N}(0, \sigma_{\text{SK}}^2)$. The noise $n_{\text{SK}}(i)$ is assumed to be independent and identically distributed (i.i.d.) for different SKs $\tilde{R}(i)$. Finally, the noise distorted SKs $R(i)$ are obtained according to Eq. (3.40):

$$R(i) = \begin{cases} s_{\min} & \text{if } \tilde{R}(i) + n_{\text{SK}}(i) < s_{\min} \\ s_{\max} & \text{if } \tilde{R}(i) + n_{\text{SK}}(i) > s_{\max} \\ \tilde{R}(i) + n_{\text{SK}}(i) & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{G} \quad (3.40)$$

Step 4:

The DK data generation is done in a similar fashion. First the noise-free DKs $\tilde{R}(i, j) \forall \{i, j\} \in \mathcal{G} : j > i$ are generated according to their hierarchical relationship classes $\alpha_{k,i,j}$ and the corresponding expected phenotype model of [BJW⁺10] as depicted in Eq. (2.13). Consequently, $\tilde{R}(i, j) = \sum_{k \in \mathcal{K}} \alpha_{k,i,j} \mu_k(i, j)$. Secondly, the noise-free SKs are corrupted by i.i.d. Gaussian noise denoted by $n_{\text{DK}}(i, j) \sim \mathcal{N}(0, \sigma_{\text{DK}}^2)$. Finally, the noise distorted DKs $R(i, j)$ are obtained according to Eq. (3.41):

$$R(i, j) = \begin{cases} s_{\min} & \text{if } \tilde{R}(i, j) + n_{\text{DK}}(i, j) < s_{\min} \\ s_{\max} & \text{if } \tilde{R}(i, j) + n_{\text{DK}}(i, j) > s_{\max} \\ \tilde{R}(i, j) + n_{\text{DK}}(i, j) & \text{otherwise} \end{cases} \quad \forall \{i, j\} \in \mathcal{G} : j > i \quad (3.41)$$

Note that the noise processes corrupting the SK and DK data, respectively, are considered to be independent.

Step 5:

Based on the DK data $R(i, j)$, the GI-profile data $\rho(i, j) \forall \{i, j\} \in \mathcal{G} : j > i$ is computed according to Eq. (2.11).

Using the procedure described above by Steps 1 – 5, the synthetic data for the MC simulations is generated with the following parameterization: Note that the two dif-

SNR values:	$\{0, 7.14, 14.29, 21.43, 28.57, 35.71, 42.86, 50\}$
MC-runs per SNR:	200
s_{\min} :	0
s_{\max} :	1
sets of genes under study \mathcal{G} :	$\{1, \dots, 5\}, \{1, \dots, 12\}$
network density:	low, high

Table 3.3. Data generation settings

ferent network density cases, i.e., *low* and *high*, have the following meaning. With the minimal number of edges in a DAG according to [BJW⁺10] being $|\mathcal{G}|$, GI-networks in the low density scenario are considered to have $1.2 \times |\mathcal{G}|$ edges in average. In the high density case, the considered GI-networks have an average number of edges of $2 \times |\mathcal{G}|$.

In Figures 3.8 to 3.15, the performance results of the proposed GENIE and GI-GENIE algorithms as well as the results of the AIS method of [BJW⁺10], described in Section 2.3, are displayed in a Monte Carlo fashion, with respect to the normalized number of erroneously estimated edges P_{ed} and the normalized number of missing edges P_{mis} , for a SNR range of 0dB to 50dB. Furthermore, let the low SNR regime be defined to comprise the region 0dB - 10dB, the intermediate SNR regime be comprised of 10dB - 40dB and the high SNR regime be defined from 40dB - 50dB.

In particular, Figures 3.8 to 3.9 show the performance results in terms of P_{ed} and P_{mis} , respectively, for a network size of $|\mathcal{G}| = 5$ genes and a low network density.

In Figure 3.8, the algorithms under study are evaluated in terms of the normalized number of erroneously estimated edges P_{ed} against the SNR. With the exception of the low SNR regime, the proposed GI-GENIE algorithm outperforms the GENIE and the AIS method over the remaining range of SNRs, showing a remarkable performance of almost 0 P_{ed} in the high SNR regime. The proposed GENIE algorithm performs

inferior to the GI-GENIE algorithm for almost the entire range of SNRs. The AIS method of [BJW⁺10], described in Section 2.3, is inferior to the proposed ILP based algorithms in terms of P_{ed} over the entire range of SNRs. Whereas the GENIE method approaches the performance of the GI-GENIE method in the high SNR regime with an error P_{ed} tending to 0, the AIS algorithm of [BJW⁺10] still performs worse than the proposed ILP based methods exhibiting an error of 20% P_{ed} which means that the number of erroneously estimated edges by the AIS method amounts to 20% of the total edges of the true DAG. In Figure 3.9, the algorithms under study are evaluated in terms of the normalized number of missing edges P_{mis} against the SNR. Here, the proposed GI-GENIE algorithm outperforms the GENIE and the AIS method over the entire range of SNRs, exhibiting a considerable performance of almost 0 P_{mis} in the high SNR regime. In a SNR range from 0dB to approximately 20dB, the proposed GENIE algorithm performs equally well compared to the AIS method of [BJW⁺10]. However, especially in the high SNR regime, the AIS method performs inferior to the GENIE algorithm showing a performance gap of 5% P_{mis} . Note that the performance gap between the GENIE algorithm and the GI-GENIE algorithm closes in the high SNR regime. Despite the mentioned performance gap between the AIS method and the proposed ILP based methods in the high SNR regime, the three investigated methods perform remarkably well in terms of P_{mis} showing that only very few edges of the true DAGs are not detected in the case of a high data quality. In summary, the proposed ILP based algorithms as well as the AIS method of [BJW⁺10], described in Section 2.3, perform considerably well in the high SNR regime in terms of P_{ed} and P_{mis} for small GI-networks of a low density.

Figures 3.10 to 3.11 show the performance results in terms of P_{ed} and P_{mis} , respectively, for a network size of $|\mathcal{G}| = 5$ genes and a high network density.

In Figure 3.10, the algorithms under study are evaluated in terms of the normalized number of erroneously estimated edges P_{ed} against the SNR. Over almost the entire SNR range, the proposed ILP based algorithms outperform the AIS method, showing a good performance of less than 10% P_{ed} in the high SNR regime. On the contrary, the AIS method of [BJW⁺10] exhibits a less impressive performance compared to the proposed ILP based methods almost stagnating in estimation quality for SNRs above 35dB. In the low SNR regime, the GENIE algorithm outperforms the GI-GENIE algorithm. In turn, the GI-GENIE algorithm performs better than the GENIE method in the intermediate SNR region. In the high SNR regime, both ILP based algorithms perform comparably well. In Figure 3.11, the considered algorithms are evaluated in terms of the normalized number of missing edges P_{mis} against the SNR. Here, the proposed GI-GENIE algorithm outperforms the GENIE and the AIS method over the entire range of SNRs. In the low and intermediate SNR regimes, the AIS method and

the GENIE method perform approximately equally well showing a good performance as well. However, in the high SNR region, the GENIE and the AIS method perform differently. While the GENIE method approaches the performance of the GI-GENIE algorithm, the AIS algorithm stagnates in error performance. In summary, the proposed ILP based algorithms perform well in terms of P_{ed} and P_{mis} , especially in the high SNR regime. On the contrary, the AIS method of [BJW⁺10], described in Section 2.3, reveals problems to estimate the true DAGs that are of a high network density. Especially in terms of the normalized number of erroneously detected edges, i.e., P_{ed} , the AIS method of [BJW⁺10] shows its limitations to reduce the number of erroneously detected edges even for a high data quality.

In the same fashion, Figures 3.12 to 3.13 show the performance results in terms of P_{ed} and P_{mis} , respectively, for a network size of $|\mathcal{G}| = 12$ genes and a low network density.

In Figure 3.12, the algorithms under study are evaluated in terms of the normalized number of erroneously estimated edges P_{ed} against the SNR. With the exception of the low SNR regime, the proposed GI-GENIE algorithm outperforms the GENIE and the AIS method over the remaining range of SNRs, showing a remarkable performance of almost 0 P_{ed} in the high SNR regime. The proposed GENIE algorithm performs inferior to the GI-GENIE algorithm for almost the entire range of SNRs, approaching the low error performance of the GI-GENIE method in the high SNR regime. On the contrary, the AIS method of [BJW⁺10], described in Section 2.3, displays a comparably high error P_{ed} over the entire range of SNRs. In Figure 3.13, the algorithms under study are evaluated in terms of the normalized number of missing edges P_{mis} against the SNR. Similar to the results presented in Figure 3.12, the GI-GENIE method outperforms the GENIE and the AIS method over the entire range of SNRs, exhibiting a considerable performance of almost 0 P_{mis} in the high SNR regime. In a SNR range from 0dB to approximately 20dB, the proposed GENIE algorithm performs equally well compared to the AIS method of [BJW⁺10]. However, especially in the high SNR regime, the AIS method performs inferior to the GENIE algorithm showing a performance gap of approximately 10% P_{mis} . Similar to the results in Figure 3.9, the performance gap between the GENIE algorithm and the GI-GENIE algorithm closes in the high SNR regime. Despite the mentioned performance gap between the AIS method and the proposed ILP based methods in the high SNR regime, the three investigated methods perform well in terms of P_{mis} showing that only few edges of the true DAGs are not detected in the case of a high data quality. In summary, the proposed ILP based algorithms perform considerably well in the high SNR regime in terms of P_{ed} and P_{mis} . In turn, the AIS method of [BJW⁺10] shows a solid performance in terms of P_{mis} , however, failing to sustainably reduce the number of erroneously estimated edges P_{ed} with the SNR increasing.

Finally, Figures 3.14 to 3.15 show the performance results in terms of P_{ed} and P_{mis} , respectively, for a network size of $|\mathcal{G}| = 12$ genes and a high network density. In Figure 3.14, the algorithms under study are evaluated in terms of the normalized number of erroneously estimated edges P_{ed} against the SNR. Over the entire SNR range, the proposed ILP based algorithms outperform the AIS method, showing a good performance of less than 10% P_{ed} in the high SNR regime. The AIS method of [BJW⁺10] shows a decreasing error P_{ed} with the SNR increasing. However, even in the high SNR regime, the number of erroneously estimated edges P_{ed} is still high with approximately 50% P_{ed} . In Figure 3.15, the considered algorithms are evaluated in terms of the normalized number of missing edges P_{mis} against the SNR. The proposed GI-GENIE algorithm shows the best performance over the entire range of SNRs. Similar to the high density network scenario with 5 genes, the AIS method and the GENIE method perform approximately equally well within the range of 0dB SNR to 28dB SNR. Interestingly, in this SNR range the AIS method performs slightly better than the GENIE algorithm. However, for high SNRs the AIS method stagnates in performance around 15% P_{mis} whereas the GENIE method approaches the performance of the GI-GENIE method.

The proposed optimization based algorithms, i.e., the GENIE method depicted in Section 3.1 as well as the GI-GENIE method depicted in Section 3.2, are based on ILP formulations of the graph learning problem. Although having a guarantee to yield optimal solutions to the graph learning problem, given the data, those algorithms suffer from a strong increase in computational cost with the GI-network size increasing. In particular, the number of binary selection variables and constraints considerably increases with the network size, while the number of candidate solutions of the proposed ILP based algorithms explodes with respect to the network size increasing.

In Figure 3.6, the number of binary selection variables and constraints of the GENIE and the GI-GENIE methods, respectively, are displayed with respect to the network size increasing. Based on Figure 3.6, it is obvious that the GI-GENIE method is computationally much more challenging than the GENIE method. As displayed in the Figure, the number of binary selection variables for both methods is already large for GI-networks of a size of 35 genes. The number of constraints of the GENIE and the GI-GENIE methods, respectively, is growing much faster with the size of the networks increasing, than the number of binary selection variables. In Figure 3.7, the numbers of *unconstrained* solution candidates of the GENIE and the GI-GENIE methods, i.e., the whole number of combinations of binary selection variables of the ILPs without considering the constraints, are exemplarily depicted for small GI-networks of sizes 4, 5 and 6. Note that the number of unconstrained solution candidates of the proposed ILP based algorithms are displayed in log-10 space for the sake of a better comparability.

In essence, the larger the GI-network, the larger the unconstrained candidate space for both methods. Furthermore, it becomes clear that the GI-GENIE algorithm is computationally much more challenging than the GENIE method, although, for both proposed methods the unconstrained solution space is still vast.

However, in light of the enormous numbers of unconstrained solution candidates, as exemplarily depicted in Figure 3.7, the large numbers of constraints of the proposed GENIE and GI-GENIE algorithms are not only guaranteeing the desired structural properties of DAGs according to [BJW⁺10]. Furthermore, in terms of mathematical tractability, they are absolutely necessary in order to considerably shrink the solution spaces of both ILP based methods. On the other hand, the beneficial functionality of the large number of constraints with respect to the size of the space of feasible solutions comes at the cost of having large optimization problems in both, numbers of variables and numbers of constraints, which extensively consumes computational resources.

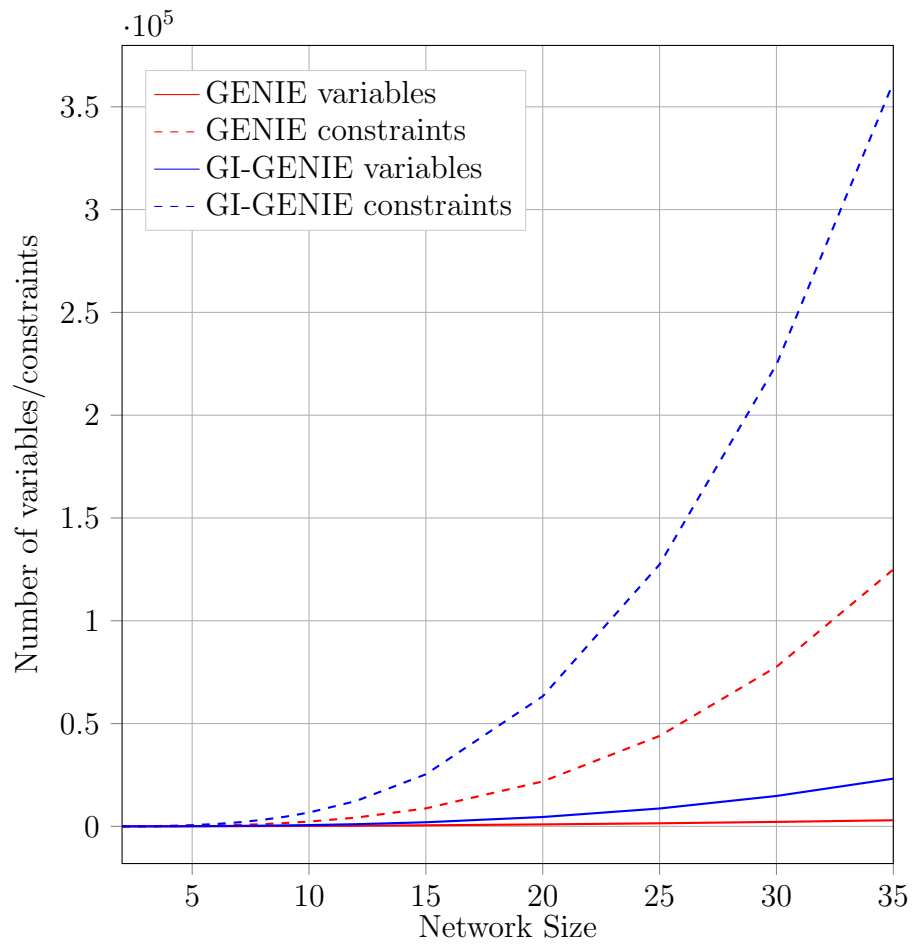


Figure 3.6. Complexity figure of the proposed ILP showing the number of binary selection variables as well as the number of constraints with respect to the network size

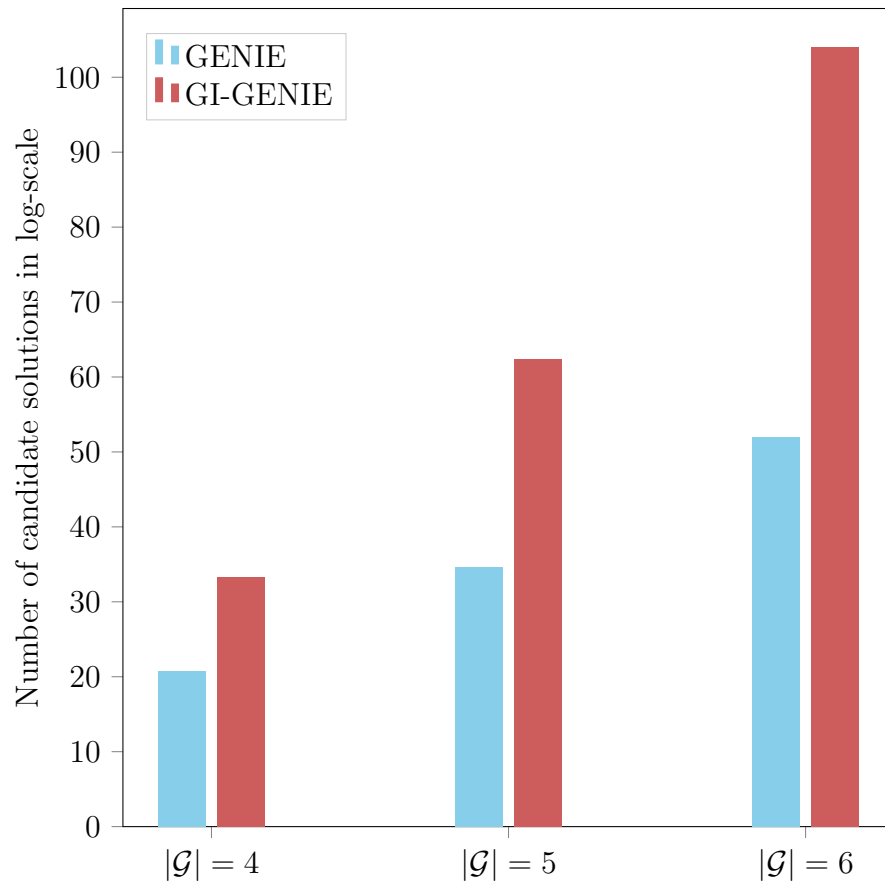


Figure 3.7. Number of *unconstrained* candidate solutions to the GENIE and the GI-GENIE problem, respectively, for selected network sizes

Elaborating on the results shown in Figures 3.8 to 3.15, as well as on the complexity Figures 3.6 and 3.7, the following observations can be made:

O_1 : Independently of the network size, the number of erroneously estimated edges P_{ed} at low SNR is higher for low density networks compared to high density networks, as shown by Figures 3.8, 3.10, 3.12 and 3.14. For low density networks the number of edges is almost equal to the number of genes in the network. Hence, such networks are (almost) minimal trees in the sense that any leaf node has only one outgoing edge, while any internal node, i.e., any gene between the leafs and the reporter node R , has exactly one outgoing edge and one incoming edge. For ideal data, this property of minimal DAGs would be reflected by the scores $s_k(i, j)$ in such a way that the scores $s_4(i, j)/s_5(i, j)$ would be very high for any pair of genes. However, for noisy data, this is not reflected at low SNRs. In particular, at low SNRs the data is corrupted to such an extent that the scores $s_k(i, j)$ indicate that there are definitely pairs of genes in classes $k = 4/k = 5$, i.e., the scores $s_4(i, j)/s_5(i, j)$ are quite low for some pairs. Indeed, this data based observation also holds for high density networks at low SNR. However, the **structural difference** between the topology estimates and the true DAGs at low SNR is bigger in the case of low density networks compared to high density networks. In the low network density case, the true DAGs $\mathcal{D}_{true, ld}$ are (almost) always a minimal DAG whereas at low SNR the topology estimates aggregated as tuple \mathbf{E}_{ld} definitely contain a considerable number of gene pairs that are classified to classes $k = 4/k = 5$. Hence, the estimates \mathbf{E}_{ld} are dense compared to the true DAGs $\mathcal{D}_{true, ld}$. In the high network density case, the true DAGs $\mathcal{D}_{true, hd}$ are dense networks, as well as the estimates aggregated by \mathbf{E}_{hd} .

At low SNR, the estimates in \mathbf{E}_{ld} and \mathbf{E}_{hd} reflect dense GI-networks. Furthermore, in the low density case the true DAGs $\mathcal{D}_{true, ld}$ are generally sparse, while in the high density case the true DAGs $\mathcal{D}_{true, hd}$ are dense. At low SNR, the difference between the true DAGs $\mathcal{D}_{true, ld}$ and the respective estimates \mathbf{E}_{ld} is larger as the difference between $\mathcal{D}_{true, hd}$ and the respective estimates \mathbf{E}_{hd} . This stems from the fact that, given a sparse ground-truth $\mathcal{D}_{true, ld}$, the more complex/dense the estimates \mathbf{E}_{ld} are, the higher is the number of erroneously detected edges. On the contrary, given a dense ground-truth $\mathcal{D}_{true, hd}$, the complex/dense estimates \mathbf{E}_{hd} are structurally more similar to the true DAGs $\mathcal{D}_{true, hd}$ which results in a comparably lower number of erroneously detected edges.

O_2 : With the SNR/data quality increasing, the above described performance difference in terms of P_{ed} due to the effect of **structural difference** vanishes.

- O_3 : For GI-networks of 5 genes and low density, the AIS method of [BJW⁺10] shows a strong performance approaching the one of the proposed GENIE algorithm. However at high SNR, for high density GI-networks of the same size the AIS algorithm according to [BJW⁺10] shows a deteriorated performance in terms of P_{ed} compared to low density networks. This stems from the fact that the initial GI-network proposal of the AIS algorithm of [BJW⁺10] described in Section 2.3 is *structurally* more different to the true DAG, which underlies the high quality data, the higher the density of the true GI-network. Due to the special structure of the proposal distribution of the AIS algorithm under the terms of [BJW⁺10], i.e., only one legal network modification summarized in Section 2.3 is proposed and not necessarily accepted at each iteration of the SA MCMC-chain, the SA sampling procedure of the AIS algorithm needs the more iterations to approach the most likely DAG reflected by the data, the more dense the true networks. In other words, given that the true DAGs are dense and the initial states/GI-networks of the SA procedure are structurally far away from the true DAGs in probability, more proposal iterations of the SA chain are needed, in order to approach the most likely DAG, reflected by the data compared to a low density true DAG.
- O_4 : Observation O_3 also holds for GI-networks of larger size as shown by Figures 3.12 and 3.14, respectively.
- O_5 : Independently of the network density, the AIS algorithm of [BJW⁺10] exhibits a worse performance in terms of P_{ed} and P_{mis} for networks of 12 genes compared to networks of 5 genes, as reflected by comparing Figures 3.8 to 3.11 with Figures 3.12 to 3.15. In general, for fixed numbers of Markov chain iterations N_{MC} and annealing runs N_{AR} , the AIS algorithm performs worse with the size of the true GI-networks increasing. This behaviour stems from the fact that the larger the GI-networks to be estimated the harder the estimation problem from a mathematical perspective. In particular, given a random initial GI-network of the AIS method of [BJW⁺10] and a true DAG underlying the data, in probability the topology difference between this initial GI-network and the true DAG is the larger the bigger the networks to be estimated. Furthermore, with the special construction of the proposal distribution of the considered AIS method, which can only propose a single network modification at a time, it becomes clear that the AIS method of [BJW⁺10] deteriorates in estimation performance with the network size increasing.
- O_6 : For a given network density scenario, the performance of the GENIE algorithm and the GI-GENIE algorithm, respectively, is similar for both network size scenarios. Whereas in the low SNR regime, the performance of the proposed GENIE

and GI-GENIE algorithms in the case of GI-networks of size 5 is better compared to the performance of those algorithms in the 12-gene GI-network case, the view changes for intermediate and high SNR regions. In the intermediate and the high SNR regime, the performance of the GENIE algorithm and the GI-GENIE algorithm, respectively, is approximately equally well for both network size scenarios. Hence, in the intermediate and high SNR regimes, the performance of the proposed ILP based algorithms introduced in Sections 3.1 and 3.2, respectively, is independent of the size of the GI-networks to be estimated.

O_7 : Independently of the network size, in high network density scenarios the GENIE and the GI-GENIE algorithms show an interesting behaviour in the high SNR regime. In particular, the performance of the GENIE method in the SNR range of 43dB and 50dB in terms of P_{ed} and P_{mis} is as good as the performance of the GI-GENIE method as depicted in Figures 3.10 and 3.11, as well as in Figures 3.14 and 3.15. This means that, for a very high data quality, the incorporation of GI-profile data into the DAG estimation procedure is not converted into a performance gain. In different words, the estimates based on multiple data types of a very high quality, i.e., the GI-GENIE DAG estimates, are as good as the estimates that are only based on one data type of very high quality, i.e., the GENIE DAG estimates. This, at the first sight, counterintuitive behaviour has its explanation in the structure of the high density DAGs. In the high network density case, the generated true DAGs also have a lot of edges that link a gene directly with the reporter node. Since the reporter node is an artificial node in the GI-network concept according to [BJW⁺10], there are neither SKs, DKs, nor GI-profile data available to assess if there is an edge between some gene and the reporter node R . In particular, the estimation of edges between genes and the reporter node R is based on the ordering condition depicted in Eq. (3.20), which is completely based on the estimated set of hierarchical relationship class selection variables $\alpha_{k,i,j}$. At very high SNRs, the information about the true DAGs in the SKs and DKs is that high, such that the information about the true DAGs contained in the GI-profile data, does not contribute any information gain with respect to the classification of the hierarchical relationship class selection variables $\alpha_{k,i,j}$. In different words, using GI-profile data together with SK and DK data in order to estimate the class selection variables $\alpha_{k,i,j}$ does not improve the quality of the estimated class selection variables $\alpha_{k,i,j}$ compared to the case that the estimation is based on SK and DK data only. Thus, the GENIE method performs as good as the GI-GENIE method at very high SNRs.

O_8 : In the high density network scenario, the performance of the proposed ILP based methods, i.e., the GENIE and the GI-GENIE algorithm, respectively, in the high

SNR regime, is inferior to the performance of those algorithms in the case of low density network scenarios, again at high SNR. This has numerical reasons. Assume that the SKs $R(i)$ and $R(j)$ of two genes i and j are very similar, i.e., $R(j) = R(i) + \delta$ with $\delta \in \mathbb{R}$ being close to zero. This can happen comparably often, since the SKs are between zero and one. In this case, the absolute difference between the expected phenotypes for classes $k = 1$ and $k = 2$ is given by $\delta_{s,1,2} = |\mu_1(i, j) - \mu_2(i, j)| = |\delta|$. Furthermore, the absolute difference between the expected phenotypes for classes $k = 4$ and $k = 5$ is given by $\delta_{s,4,5} = |\mu_4(i, j) - \mu_5(i, j)| = \frac{1}{2}|\delta|$.

In the **low density scenarios** at high SNR, given the data it is very rare that two genes are in classes $k = 4$ or $k = 5$, respectively. In other words, the mismatch scores $s_1(i, j)$, $s_2(i, j)$ and $s_3(i, j)$ are much smaller than scores $s_4(i, j)$ and $s_5(i, j)$. The estimated set of classes will be dominated by classes $k = 1$, $k = 2$ and $k = 3$. As shown above the decision whether to classify a pair of genes $\{i, j\}$ into classes $k = 1$ or $k = 2$ can be very hard, since the expected phenotypes $\mu_1(i, j)$ and $\mu_2(i, j)$ are very close. Hence, misclassifications between classes $k = 1$ and $k = 2$ can happen even in the case of (almost) perfect data. In the **high density scenarios** at high SNR, given the data it occurs quite frequently that two genes are in classes $k = 4$ or $k = 5$, respectively. The estimated set of classes will be dominated by classes $k = 4$ and $k = 5$. As shown above the decision whether to classify a pair of genes $\{i, j\}$ into classes $k = 4$ or $k = 5$ can be even harder, since the expected phenotypes $\mu_4(i, j)$ and $\mu_5(i, j)$ can be even closer. Thus, misclassifications between classes $k = 4$ and $k = 5$ can happen even in the case of (almost) perfect data.

In summary, in the low density case at high SNR, the classification challenge is to distinguish between classes $k = 1$ and $k = 2$, whereas in the high density case at high SNR, the challenge is given by differentiating between the classes $k = 4$ and $k = 5$. Since differentiating between classes $k = 4$ and $k = 5$ is numerically more challenging in average than distinguishing between classes $k = 1$ and $k = 2$, the errors P_{ed} and P_{mis} are higher for high density networks compared to low density networks, again at high SNR.

In essence, the following summarizing conclusions can be drawn based on the performance results depicted in Figures 3.8 to 3.15 as well as on the observations O_1 to O_8 stated above.

With the given parameterization, the AIS method of [BJW⁺10] yields good estimation results in terms of P_{ed} and P_{mis} for low density networks of small size. However, with the

network size and the network density increasing, the presented AIS method deteriorates in estimation performance, especially for the task of estimating large GI-networks. In order to obtain better estimation results in the case of large networks or high density networks, the number of SA Markov-chain iterations N_{MC} , as well as the number of annealing runs N_{AR} can be increased. The so caused performance gain comes at the cost of a higher computational cost which makes the inherently slow MCMC idea behind the AIS method even slower. Furthermore, it is not known a priori how many iterations are needed in order to obtain estimation results of a sufficient quality. However, despite their being inherently slow, MCMC based algorithms like the AIS method of [BJW⁺10] do not suffer from a *combinatorial explosion* like convex or discrete optimization based techniques.

The proposed ILP based methods depicted in Sections 3.1 and 3.2, i.e., the GENIE and the GI-GENIE algorithm, respectively, show strong performance results in terms of P_{ed} and P_{mis} in general. In particular, at intermediate and high SNR the error performance of the proposed GENIE and GI-GENIE methods is independent of the size of the underlying GI-network which is to be estimated. Although exhibiting a slightly worse performance in the case of high density networks compared to the case of low density networks, the GENIE and the GI-GENIE algorithms still perform very well. Especially in the situation of sufficient/high data quality, i.e., in the intermediate and high SNR regimes, the proposed ILP based algorithms performance is approximately independent of the underlying network density, taking into account the numerical issues discussed in observation O_8 .

Furthermore, based on the performance results displayed in Figures 3.8 to 3.15, it has been shown that the incorporation of multiple data types into the estimation procedure, as in the GI-GENIE algorithm, is superior to the case when only SKs/DKs are considered for in the estimation procedure, as in the GENIE algorithm. However, the increased performance of the GI-GENIE algorithm with respect to the GENIE algorithm comes at the cost of a much higher computational complexity. Although the proposed ILP based methods perform independently of the network size in terms of P_{ed} and P_{mis} at intermediate and high SNR, the GENIE as well as the GI-GENIE are not completely unaffected by the network size. With the network size increasing, the computational cost increases exponentially. Due to the combinatorial explosion of the feasible sets of the GENIE and the GI-GENIE as depicted in Figures 3.6 and 3.7, respectively, those algorithms are computationally tractable only for moderate network sizes if no large computation clusters are available.

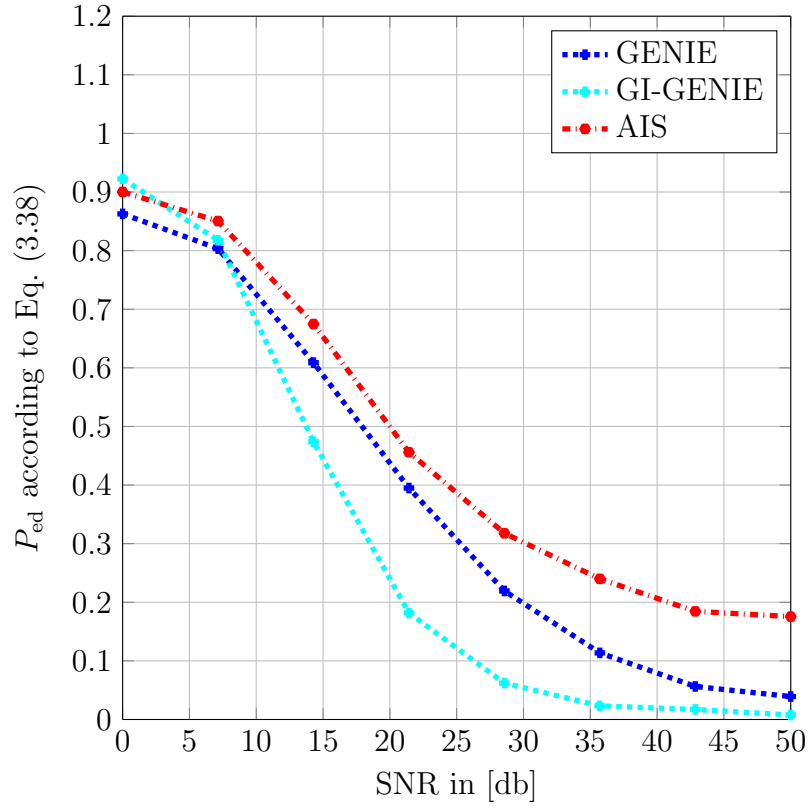


Figure 3.8. Normalized number of erroneously detected edges according to Eq. (3.38) vs. SNR; network/DAG size $|\mathcal{G}| = 5$; low density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{AR} = 100$, SA-iterations $N_{MC} = 100$

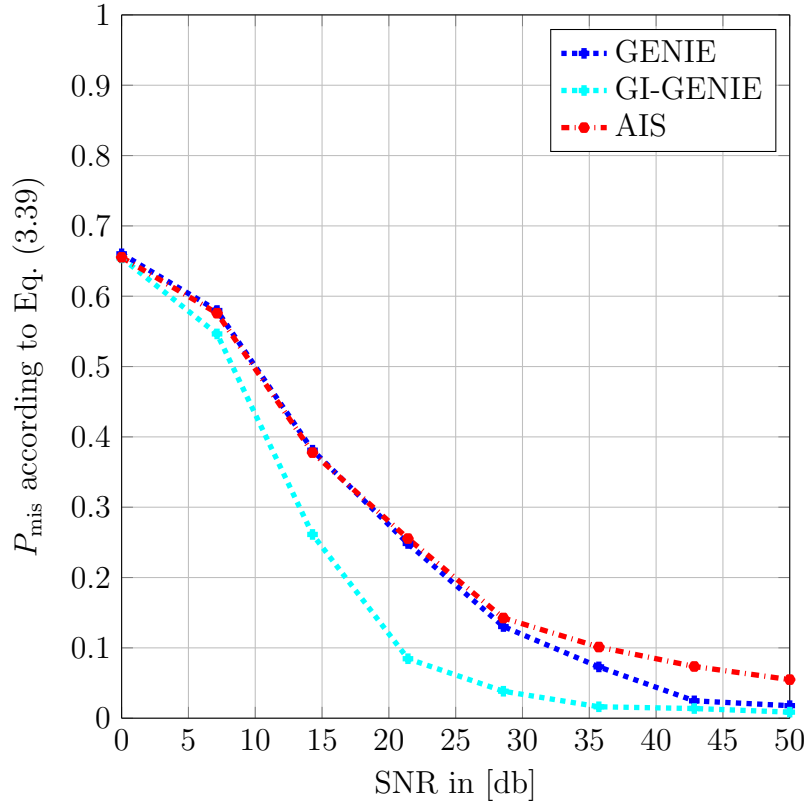


Figure 3.9. Normalized number of missing edges according to Eq. (3.39) vs. SNR; network/DAG size $|\mathcal{G}| = 5$; low density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{\text{AR}} = 100$, SA-iterations $N_{\text{MC}} = 100$

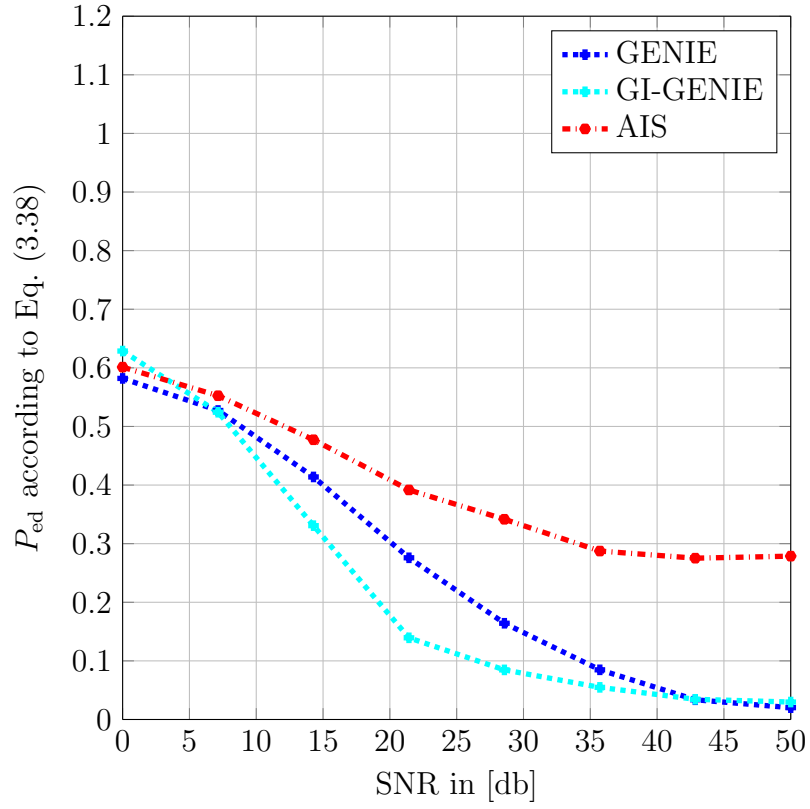


Figure 3.10. Normalized number of erroneously detected edges according to Eq. (3.38) vs. SNR; network/DAG size $|\mathcal{G}| = 5$; high density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{AR} = 100$, SA-iterations $N_{MC} = 100$

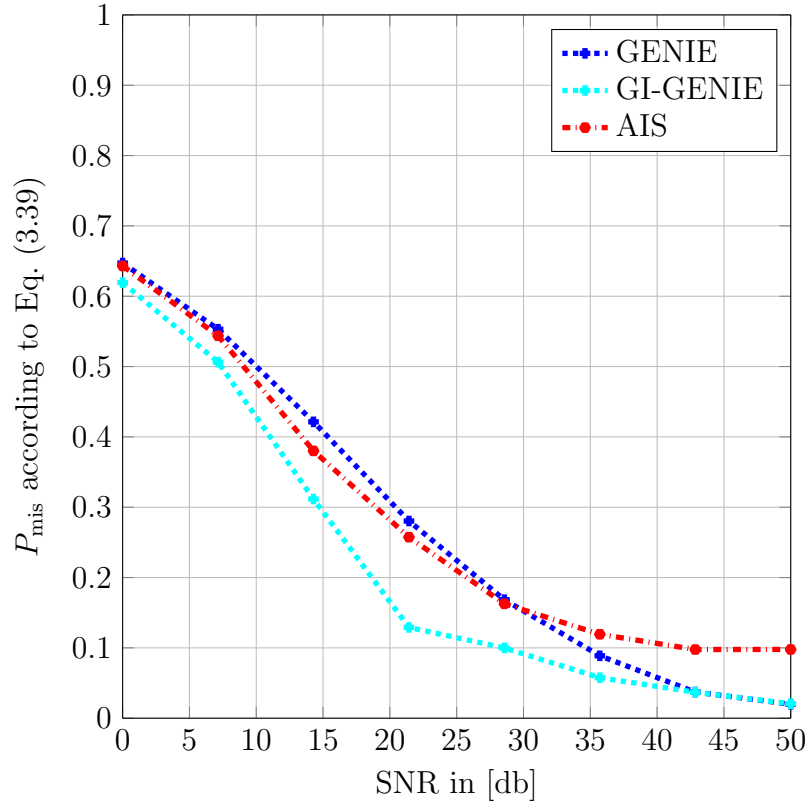


Figure 3.11. Normalized number of missing edges according to Eq. (3.39) vs. SNR; network/DAG size $|\mathcal{G}| = 5$; high density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{\text{AR}} = 100$, SA-iterations $N_{\text{MC}} = 100$

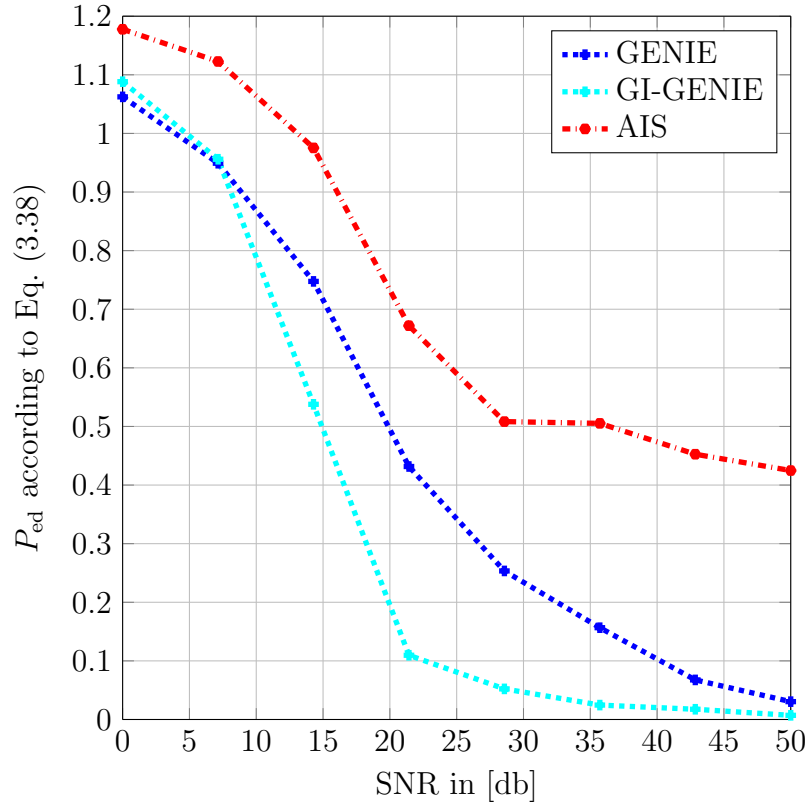


Figure 3.12. Normalized number of erroneously detected edges according to Eq. (3.38) vs. SNR; network/DAG size $|\mathcal{G}| = 12$; low density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{AR} = 100$, SA-iterations $N_{MC} = 100$

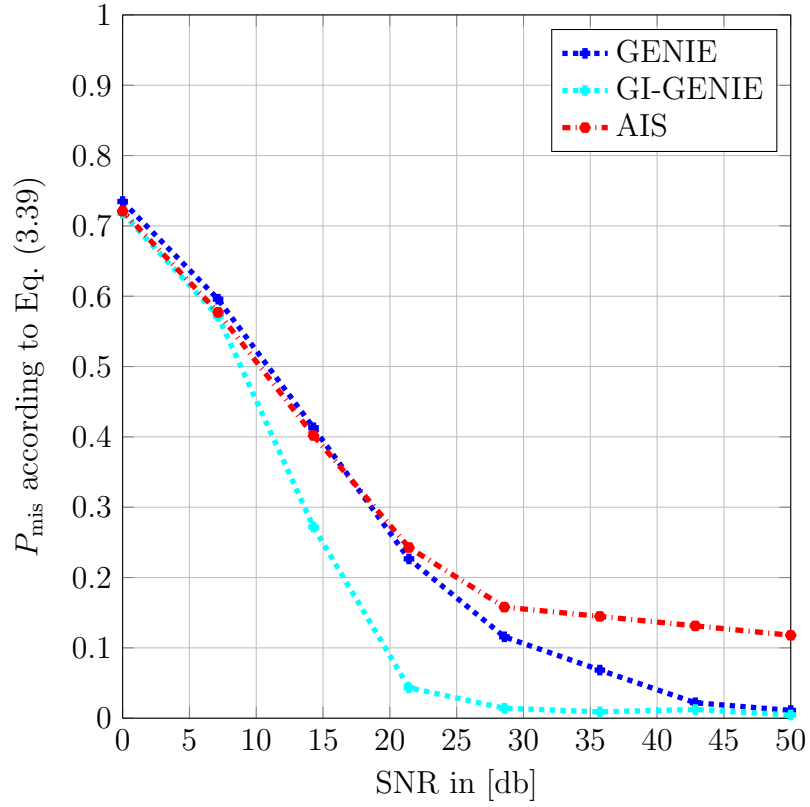


Figure 3.13. Normalized number of missing edges according to Eq. (3.39) vs. SNR; network/DAG size $|\mathcal{G}| = 12$; low density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{\text{AR}} = 100$, SA-iterations $N_{\text{MC}} = 100$

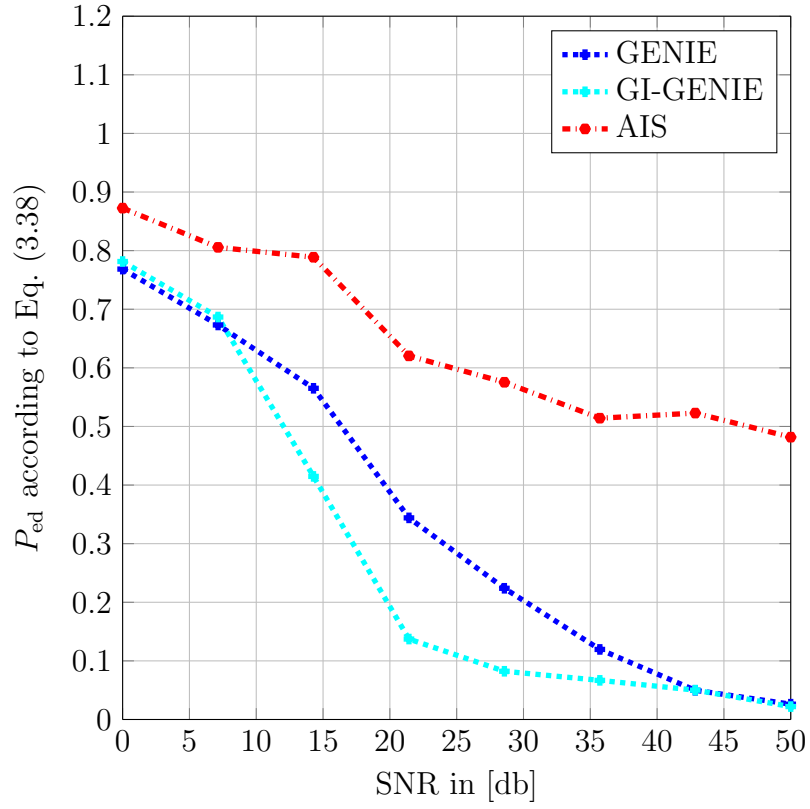


Figure 3.14. Normalized number of erroneously detected edges according to Eq. (3.38) vs. SNR; network/DAG size $|\mathcal{G}| = 12$; high density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{\text{AR}} = 100$, SA-iterations $N_{\text{MC}} = 100$

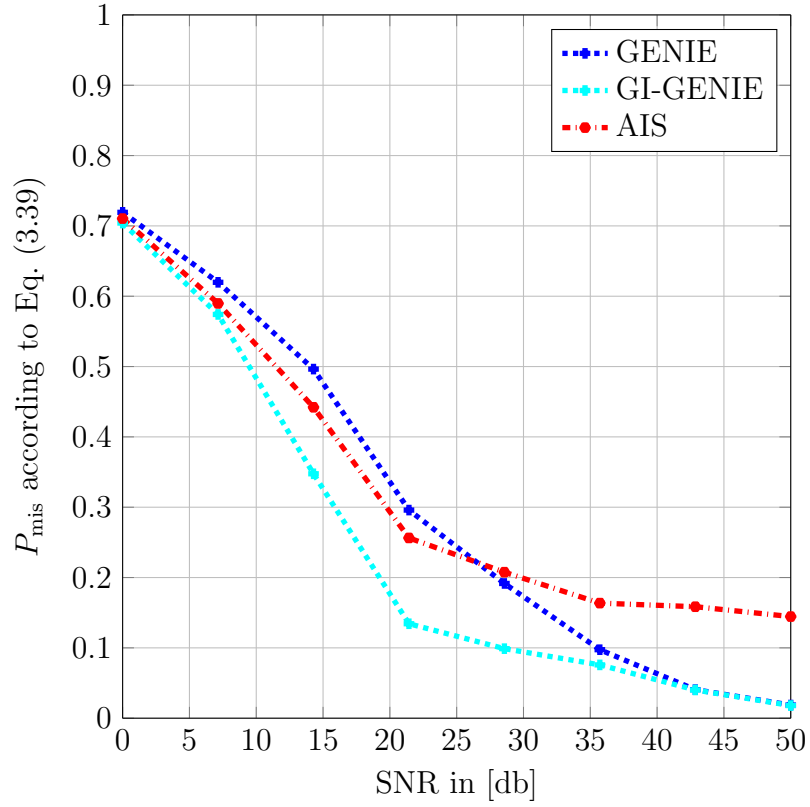


Figure 3.15. Normalized number of missing edges according to Eq. (3.39) vs. SNR; network/DAG size $|\mathcal{G}| = 12$; high density; 200-MC runs per SNR value; GI-GENIE parameters: $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.75$; AIS parameters: annealing runs $N_{\text{AR}} = 100$, SA-iterations $N_{\text{MC}} = 100$

3.4.2 Real Data Results

In this subsection, the SEQSCA algorithm proposed in Section 3.3 using the GENIE, GI-GENIE and AIS algorithms, respectively, is applied to the large-scale dataset of $[C^+]$ in order to compute the corresponding reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} and \mathbf{M}_{AIS} which reflect GI-networks of simplified gene-gene interactions weighted with their empirical probability.

In particular, the SK and DK phenotypes as well as the GI-profile data reported in $[C^+]$ have been generated by large-scale knockout experiments conducted on yeast. In those experiments, the cell function under study is “bacteria growth” that serves as a proxy for “bacteria fitness”. Typically, the bacteria growth is measured in terms of colony size, which is normalized to the colony size of unmutated bacteria colonies, i.e., to the wild type (WT). Due to the fixed-size Petri dishes where the yeast colonies are grown, the size of the yeast colonies in $[C^+]$, i.e., the particular values of the SK/DK phenotypes, is situated between 0 and a maximum value s_{\max} that is dictated by the Petri dish size and the size of the WT. Hence, the SK/DK phenotypes in $[C^+]$ are non-negative real numbers within the range $[0, s_{\max}]$. The *query genes list* of $[C^+]$ contains all genes that have been considered in this knockout experiment. However, for computational reasons only the first 200 genes that list are considered. Thus, in this subsection the set of genes under study, i.e., \mathcal{G} , is composed of the first 200 genes of the *query genes list* of $[C^+]$.

Since discovering genetic interactions, i.e., GI-networks/DAGs, for specific organisms is an ongoing field of research and the knowledge on genetic interactions is far away from being complete, there is generally no *ground-truth* to directly compare with, even not for yeast which is one of the best understood organisms in this aspect.

Therefore, the quality assesement of the learned reliability matrices, i.e., \mathbf{M}_G , \mathbf{M}_{GI} and \mathbf{M}_{AIS} , is based on the following two aspects:

Aspect 1:

In systems biology it is generally known that genetic interactions are rare. Thus, the unknown true reliability matrix \mathbf{M}_{true} for the given data set \mathcal{G} is sparse in a sense that the overwhelming number of interactions/entries in \mathbf{M}_{true} are 0 and only a very few are 1. Hence, $\mathbf{M}_{\text{true}} \in \{0, 1\}^{200 \times 200}$. Therefore, a reasonable estimate of \mathbf{M}_{true} approaches sparsity in the sense that the very most of the considered gene pairs interact with an emperical probability that is very small, a few considered gene pairs interact with an emperical probability that is very high. Furthermore, the

number of considered gene pairs, that interact with an empirical probability of around 50% should also be small. Note that in this sense, there can be many equally well performing reliability matrices having quite different structures.

Aspect 2:

There are research projects in systems biology that aim at aggregating as much genetic information about some organism as possible in order to provide a database of knowledge about that specific organism. For yeast, there is a genome database online freely accessible under [yea]. This database aggregates the knowledge with respect to genetic interactions of a plethora of knockout experiments. Furthermore, there is also an abundance of hand- curated genetic interactions knowledge of experts based on different kinds of biological and chemical experiments. Therefore, the number of highly confident interactions present in the different reliability matrices, which are also reported in the database of [yea], can be seen as one kind of quality measure of the respective reliability matrices. As a general rule, the more highly confident interactions, which are also reported in [yea], a reliability matrix has, the better the estimation quality.

However, it is important to remark again that the knowledge in [yea] is far from being complete and cannot be seen as a ground truth for assessing the estimation quality of the learned reliability matrices. Consequently, the database in [yea] can be rather seen as a helpful source of partial evidence assisting in the evaluation of the learned reliability matrices.

In Figures 3.16 to 3.18, the reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} and \mathbf{M}_{AIS} are displayed that have been computed with the SEQSCA-method using the GENIE-algorithm, the SEQSCA-method using the GI-GENIE-algorithm and the SEQSCA-method using the AIS-algorithm of [BJW⁺10], respectively. For all of the presented results in Figures 3.16 to 3.18, the SEQSCA-method divided the large set of genes \mathcal{G} into a sequence of $S = 5e^4$ subsets each of size $N_S = 10$.

Sparsity in terms of Aspect 1:

In order to evaluate the reliability matrices displayed in Figures 3.16 to 3.18 in terms of sparsity according to aspect 1, the thresholds ϵ_{low} and ϵ_{high} are introduced. Any entry of \mathbf{M}_G , \mathbf{M}_{GI} and \mathbf{M}_{AIS} , respectively, that is below ϵ_{low} is considered to represent a gene pair that interacts with a “lower empirical probability”. On the contrary, any entry of the considered reliability matrices that is above ϵ_{high} is assumed to represent a gene pair that interacts with a “higher empirical probability”. Let the lower threshold ϵ_{low} be 0.3 and the higher threshold ϵ_{high} be 0.7, i.e., $\epsilon_{\text{low}} = 0.3$ and $\epsilon_{\text{high}} = 0.7$.

With ϵ_{low} and ϵ_{high} as defined above, 87% of the considered genetic interactions in the reliability matrix \mathbf{M}_G in Figure 3.16 either interact with an empirical probability that is lower than ϵ_{low} or higher than ϵ_{high} . In other words, the decisions about the interactions of the considered gene pairs are uncertain only in 13% of all cases. Furthermore, 77% of the considered gene-pairs in \mathbf{M}_G interact with an empirical probability that is lower than ϵ_{low} . On the contrary, only 11% of the considered gene-pairs in \mathbf{M}_G interact with an empirical probability that is higher than ϵ_{high} . Hence, the performance of the SEQSCA using the GENIE algorithm is quite well in terms of aspect 1.

In the case of the SEQSCA method utilizing the GI-GENIE algorithm, 95% of the considered genetic interactions in the reliability matrix \mathbf{M}_{GI} in Figure 3.17 either interact with an empirical probability that is lower than ϵ_{low} or higher than ϵ_{high} . Consequently, only 5% of the considered gene pairs interact with a chance of approximately 50%, i.e., the uncertainty about the very most gene pairs is very low. In particular, 87% of the considered gene pairs in \mathbf{M}_{GI} interact with an empirical probability that is lower than ϵ_{low} and 8% of the considered gene pairs interact with an empirical probability that is higher than ϵ_{high} . In essence, the combination of the SEQSCA method with the GI-GENIE algorithm shows an impressive performance in terms of sparsity according to aspect 1.

The reliability matrix \mathbf{M}_{AIS} , that stems from the combination of the SEQSCA method and the AIS algorithm of [BJW⁺10], exhibits that 71% of the considered genetic interactions in the reliability matrix \mathbf{M}_{AIS} in Figure 3.18 either interact with an empirical probability that is lower than ϵ_{low} or higher than ϵ_{high} . In addition, 71% of the considered gene pairs in \mathbf{M}_{AIS} interact with an empirical probability that is lower than ϵ_{low} . Furthermore, only 1% of the considered gene pairs in \mathbf{M}_{AIS} interact with an empirical probability that is higher than ϵ_{high} . Hence, in terms of sparsity according to aspect 1, the combination of the SEQSCA algorithm with the AIS method of [BJW⁺10] yields solid results, although it falls behind in performance compared to the SEQSCA method using the GENIE and GI-GENIE algorithms, respectively. Moreover, the combination of the SEQSCA algorithm and the AIS method interestingly shows only a very few gene pairs that interact with an empirical probability of 70% or higher. Thus, SEQSCA together with the AIS method of [BJW⁺10] can be seen as a very “conservative” estimator in a sense that only in the presence of strong evidence in the data, a pair of genes is considered to interact with a higher empirical probability, i.e., an empirical probability higher than ϵ_{high} .

Performance in terms of Aspect 2:

Let $\Gamma(\epsilon)$ denote the acceptance ratio of confidence ϵ and be defined according to

Eq. (3.42)

$$\Gamma(\epsilon) = \frac{N_r(\epsilon)}{N_t(\epsilon)} \quad (3.42)$$

where $N_r(\epsilon)$ denotes the number of gene pairs in a reliability matrix which interact with an empirical probability/confidence that is higher $1 - \epsilon$ and $N_t(\epsilon)$ denotes the fraction of $N_r(\epsilon)$ which is also reported in [yea]. Hence, the acceptance ratio $\Gamma(\epsilon)$ with respect to a confidence level ϵ measures a reliability matrix's performance in terms of aspect 2. In Table 3.4, the acceptance ratios according to Eq. (3.42) with respect to $\epsilon = 0.3$ and

Method:		$\epsilon = 0.3$	$\epsilon = 0.1$
SEQSCA & GENIE	Γ :	6.7%	11%
SEQSCA & GI-GENIE	Γ :	9%	14%
SEQSCA & AIS	Γ :	4.8%	0%

Table 3.4. Acceptance ratios

$\epsilon = 0.1$ are displayed for the reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} and \mathbf{M}_{AIS} , respectively. Hence, the acceptance ratios of Table 3.4 are based on “confident/highly confident” interactions only. According to Table 3.4 and for $\epsilon = 0.3$, 6.7% of the interactions in \mathbf{M}_G , which are more likely than 70%, are also reported in the database of [yea]. For $\epsilon = 0.3$ and the SEQSCA method using the GI-GENIE algorithm, 9% of the interactions in \mathbf{M}_{GI} , that are more likely than 70%, are also reported in the database of [yea] which emphasizes again that the combination of multiple data types, i.e., SK/DK data and GI-profile data, is beneficial in terms of estimation performance. Furthermore, for $\epsilon = 0.3$ and the SEQSCA method using the AIS algorithm of [BJW⁺10], 4.8% of the interactions in \mathbf{M}_{AIS} , that are more likely than 70%, are also reported in the database of [yea].

The acceptance ratios for $\epsilon = 0.1$ are only based on genetic interactions that have been found in the corresponding reliability matrices with very high confidence, i.e., only interactions more likely than 90% have been considered. In this case, 11% of the interactions in \mathbf{M}_G , which are more likely than 90%, are also reported in the database of [yea]. For the SEQSCA algorithm using the GI-GENIE method, 14% of the interactions in \mathbf{M}_{GI} , which are more likely than 90%, are also reported in the database of [yea]. Finally, for the SEQSCA algorithm using the AIS method of [BJW⁺10], no interaction in \mathbf{M}_{AIS} has been estimated with more than 90% of empirical probability. Hence, the acceptance ratio in this case is 0%. The results displayed in Table 3.4 essentially show that the higher the empirical interaction probability of a gene pair in the computed reliability matrices, the more likely this specific interaction is also reported in [yea]. At this point, it is important to remark, that although the acceptance

ratios in Table 3.4 might appear low the investigated algorithms' performance is still good. As mentioned above, the database of [yea] aggregates the knowledge of a plethora of different knockout experiments and biochemical experiments. Hence, the interactions reported in [yea] are based on a very rich data record, whereas for this dissertation only the data of [C⁺] has been considered. Thus, in the light of this data record disadvantage, the acceptance ratios in presented in Table 3.4 are still solid performance results.

Similarity evaluation of the estimates:

Finally, one can compare the reliability matrices among each other in order to assess how similar their results are. Let $\tilde{\mathbf{\Gamma}}(\epsilon) = (\tilde{\Gamma}(\epsilon)_{\text{low}}, \tilde{\Gamma}(\epsilon)_{\text{inter.}}, \tilde{\Gamma}(\epsilon)_{\text{high}})$ denote the similarity measures for two arbitrary reliability matrices denoted as \mathbf{M}_1 and \mathbf{M}_2 with respect to the confidence ϵ . In particular, the similarity measures $(\tilde{\Gamma}(\epsilon)_{\text{low}}, \tilde{\Gamma}(\epsilon)_{\text{inter.}}$ and $\tilde{\Gamma}(\epsilon)_{\text{high}})$ are defined according to Eq. (3.43)

$$\tilde{\Gamma}(\epsilon)_{\text{low}} = \frac{|\mathcal{N}_{\text{low},1} \cap \mathcal{N}_{\text{low},2}|}{|\mathcal{N}_{\text{low},1} \cup \mathcal{N}_{\text{low},2}|}, \quad \tilde{\Gamma}(\epsilon)_{\text{inter.}} = \frac{|\mathcal{N}_{\text{inter},1} \cap \mathcal{N}_{\text{inter},2}|}{|\mathcal{N}_{\text{inter},1} \cup \mathcal{N}_{\text{inter},2}|}, \quad \tilde{\Gamma}(\epsilon)_{\text{high}} = \frac{|\mathcal{N}_{\text{high},1} \cap \mathcal{N}_{\text{high},2}|}{|\mathcal{N}_{\text{high},1} \cup \mathcal{N}_{\text{high},2}|} \quad (3.43)$$

where $\mathcal{N}_{\text{low},1}/\mathcal{N}_{\text{low},2}$ denote the sets of gene pairs $\{i, j\}$ of reliability matrices \mathbf{M}_1 and \mathbf{M}_2 , respectively, for which the empirical interaction probability is lower than ϵ , i.e., $M_1(i, j) < \epsilon/M_2(i, j) < \epsilon$. Hence,

$$\mathcal{N}_{\text{low},1} = \left\{ \{i, j\} : M_1(i, j) < \epsilon \right\} \quad \mathcal{N}_{\text{low},2} = \left\{ \{i, j\} : M_2(i, j) < \epsilon \right\}. \quad (3.44)$$

In a similar fashion $\mathcal{N}_{\text{inter},1}/\mathcal{N}_{\text{inter},2}$ as well as $\mathcal{N}_{\text{high},1}/\mathcal{N}_{\text{high},2}$ are defined as

$$\mathcal{N}_{\text{inter},1} = \left\{ \{i, j\} : M_1(i, j) \geq \epsilon \wedge M_1(i, j) \leq 1 - \epsilon \right\} \quad (3.45)$$

$$\mathcal{N}_{\text{inter},2} = \left\{ \{i, j\} : M_2(i, j) \geq \epsilon \wedge M_2(i, j) \leq 1 - \epsilon \right\} \quad (3.46)$$

and

$$\mathcal{N}_{\text{high},1} = \left\{ \{i, j\} : M_1(i, j) > 1 - \epsilon \right\} \quad \mathcal{N}_{\text{high},2} = \left\{ \{i, j\} : M_2(i, j) > 1 - \epsilon \right\}. \quad (3.47)$$

In essence, the similarity measures defined in Eq. (3.43) measure how many of the gene pairs, that interact with low/intermediate/high empirical probability, have two arbitrary reliability matrices in common.

In Table 3.5, the reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} and \mathbf{M}_{AIS} are compared with each other based on the similarity measures $\tilde{\mathbf{\Gamma}}(\epsilon)$ defined in Eq. (3.43).

GENIE vs. GI-GENIE					AIS vs. GENIE					AIS vs. GI-GENIE				
similarity $\tilde{\Gamma}(\epsilon = .3)$:					similarity $\tilde{\Gamma}(\epsilon = .3)$:					similarity $\tilde{\Gamma}(\epsilon = .3)$:				
	low	inter.	high			low	inter.	high			low	inter.	high	
	88%	21%	64%			61%	9%	1%			67%	4%	2%	

Table 3.5. Similarity comparison of reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} and \mathbf{M}_{AIS} that correspond the to GENIE method, the GI-GENIE method and the AIS method of [BJW⁺10], respectively.

The intuition that the results displayed by the reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} as depicted in Figures 3.16 and 3.17, respectively, are similar, is objectively emphasised by the respective similarity measures $\tilde{\Gamma}(\epsilon)$ displayed in Table 3.5. In particular, 88% of the gene pairs, which are in the *low probability regime* according to \mathbf{M}_G or \mathbf{M}_{GI} , i.e., they interact with an empirical probability of equal to or less than 30% ($\epsilon = .3$), are also in the low probability regime in both reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} . Furthermore, 64% of the gene pairs, which are in the *high probability regime* according to \mathbf{M}_G or \mathbf{M}_{GI} , i.e., they interact with an empirical probability of equal to or more than 70% ($1 - \epsilon = .7$), are also in the high probability regime in both reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} . Finally, 21% of the gene pairs of the *intermediate probability regime* according to \mathbf{M}_G or \mathbf{M}_{GI} , i.e., the empirical interaction probability of those gene pairs is between 30% and 70%, are also in the intermediate probability regime in both reliability matrices \mathbf{M}_G , \mathbf{M}_{GI} .

Since the GENIE and the GI-GENIE are similar, but not identical algorithms, their reliability matrices \mathbf{M}_G and \mathbf{M}_{GI} , respectively, should be similar as well. Exactly this is demonstrated by the similarity measures displayed in Table 3.5 which emphasize that the GENIE and the GI-GENIE based reliability matrices, i.e., \mathbf{M}_G and \mathbf{M}_{GI} , respectively, are structurally very similar. Note that the reliability matrices \mathbf{M}_G and \mathbf{M}_{GI} cannot agree to 100% with respect to the similarity measures $\tilde{\Gamma}(\epsilon)$, since the GI-GENIE based reliability matrix \mathbf{M}_{GI} shows different (better) results in terms of *aspect 1* and *aspect 2* compared to the GENIE based reliability matrix \mathbf{M}_G , as elucidated above in detail.

The detailed comparison between the AIS based reliability matrix and the GENIE and the GI-GENIE bases reliability matrices, respectively, can be done in the same fashion as the comparison between the GENIE and the GI-GENIE based reliability matrices. Ultimately, it is important to remark that the intuitively recognized large difference between \mathbf{M}_{AIS} and \mathbf{M}_G as well as between \mathbf{M}_{AIS} and \mathbf{M}_{GI} , is objectively justified by the respective similarity measures depicted in Table 3.5.

Neither the GENIE based reliability matrix \mathbf{M}_G nor the GI-GENIE based reliability matrix \mathbf{M}_{GI} share many gene pairs out of their intermediate/high empirical interaction probability regime with the respective empirical interaction probability regime of the AIS based reliability matrix \mathbf{M}_{AIS} . The relatively high agreement between \mathbf{M}_{AIS} and \mathbf{M}_G as well as between \mathbf{M}_{AIS} and \mathbf{M}_{GI} in the low empirical probability regime stems from the fact that the very most of the gene pairs under study interact with a low empirical probability according to all of the three investigated methods.

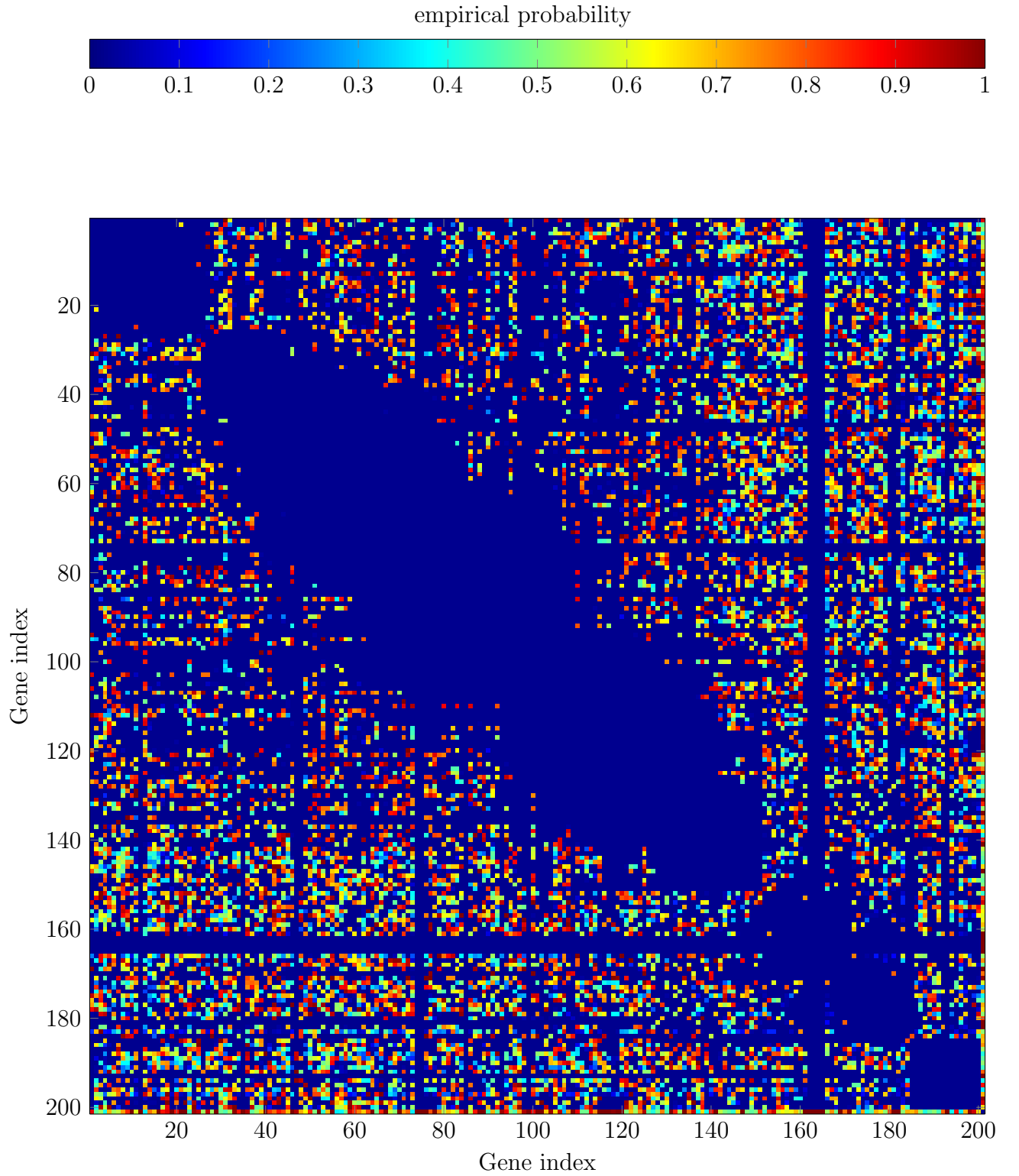


Figure 3.16. SEQSCA & GENIE: reliability matrix \mathbf{M}_G ; $S = 5e^4$ subsets considered; subset size $N_S = 10$

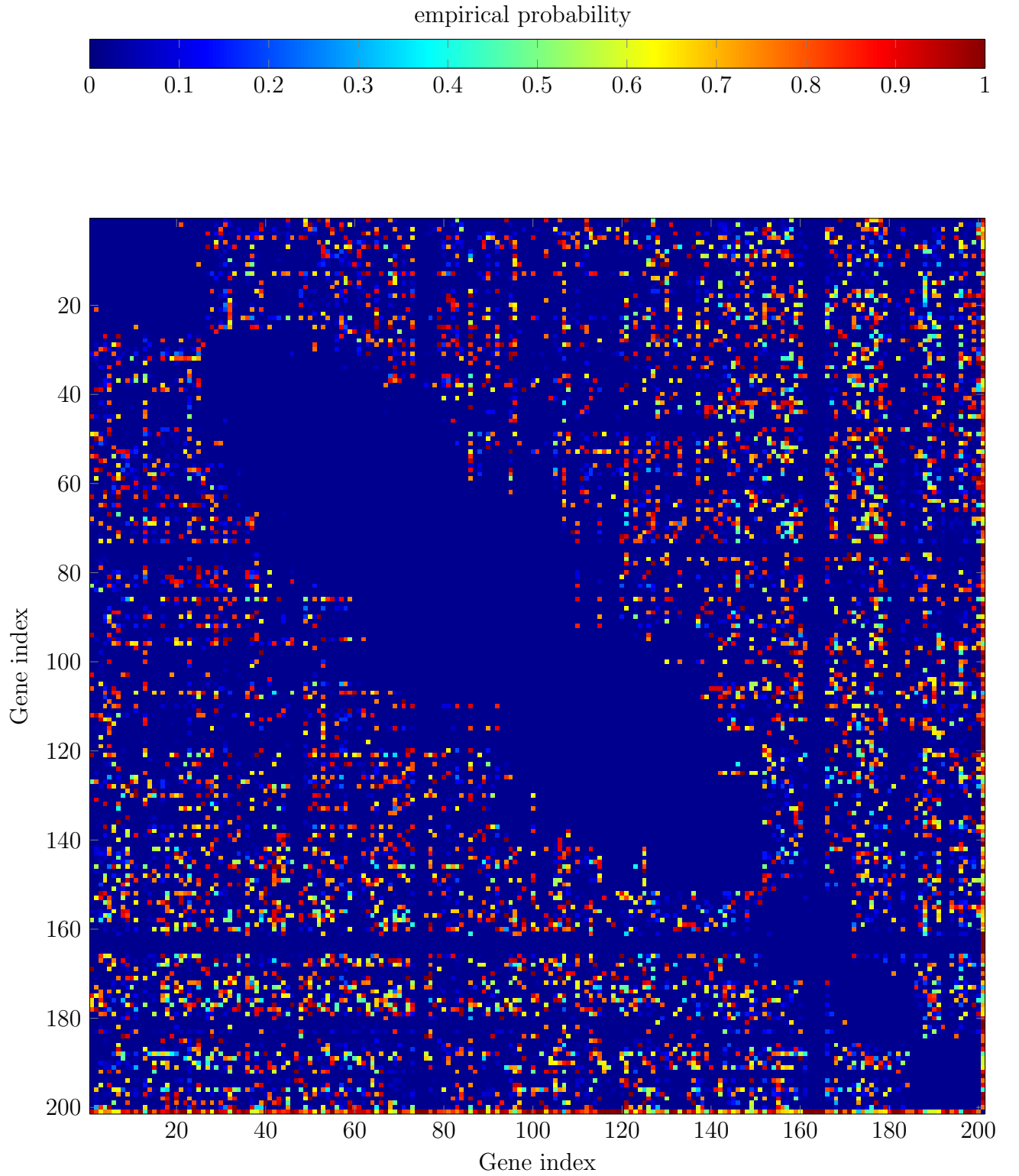


Figure 3.17. SEQSCA & GI-GENIE: reliability matrix \mathbf{M}_{GI} ; $S = 5e^4$ subsets considered; subset size $N_S = 10$; $\lambda_d = 1e3$, $\lambda_c = 1$, $\lambda_p = 0.85$

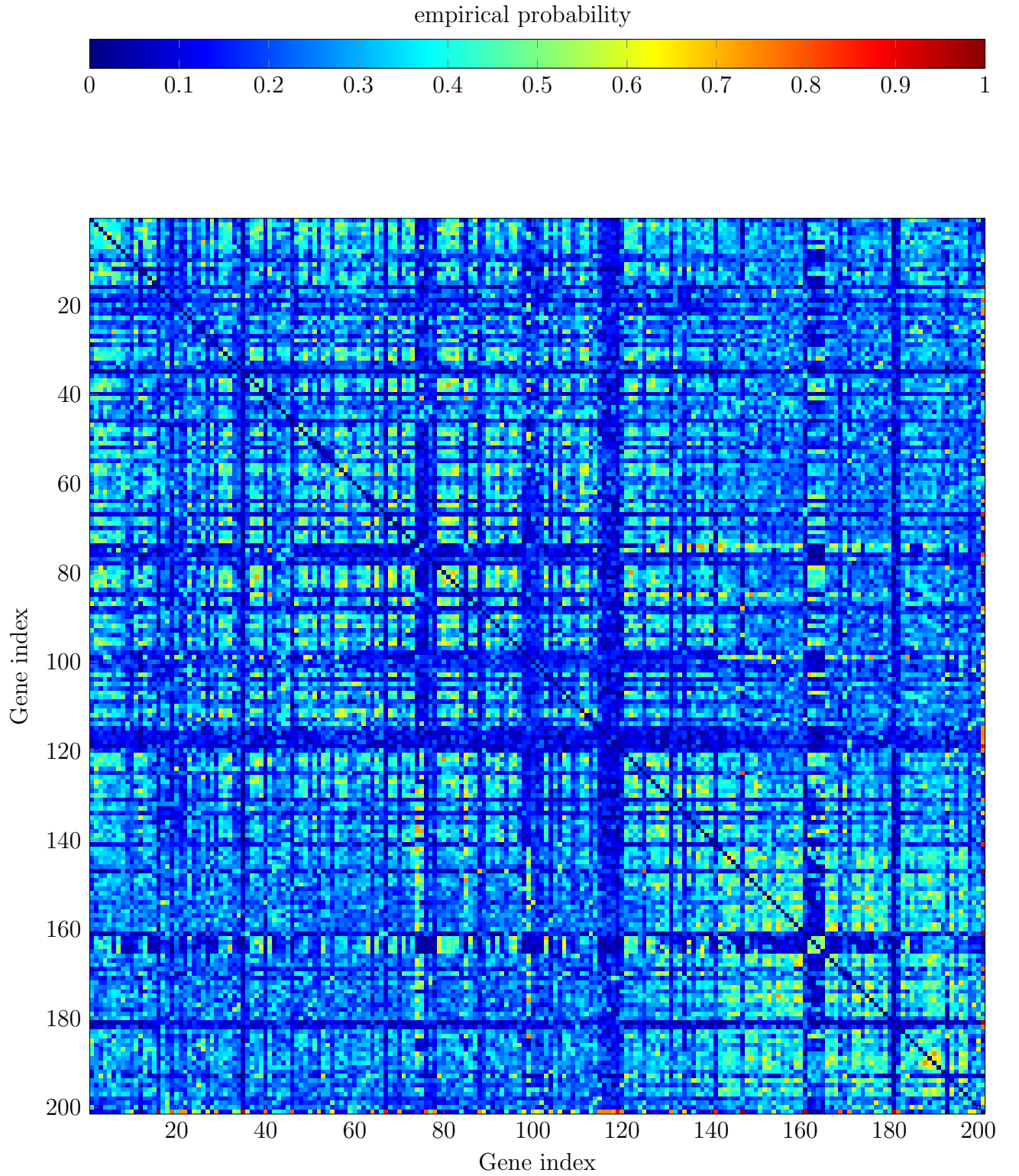


Figure 3.18. SEQSCA & AIS: reliability matrix \mathbf{M}_{AIS} ; $S = 5e^4$ subsets considered; subset size $N_S = 10$; annealing runs $N_{\text{AR}} = 100$; MCMC-transitions $N_{\text{TR}} = 100$

3.5 Summary

In this chapter, ILP formulations in order to learn the DAG, underlying SK/DK and GI-profile data, have been proposed.

In particular, in Section 3.1 the GENIE algorithm has been presented which estimates the DAG topology in a two stage procedure based on SK and DK data only. First, the GENIE problem in (3.5), which is an ILP, is solved in order to obtain the hierarchical class selection variables that describe the DAG in a different domain. Note that the GENIE problem of (3.5) can be solved efficiently using BB and BC methods that are embedded in commercial solvers as CPLEX [Flo95], Gurobi [Don] and Mosek [BV11]. In the second stage, the estimated hierarchical class selection variables are used to recover the actual DAG topology according to the procedure described in Algorithm 4.

Furthermore, in Section 3.2 the GI-GENIE algorithm has been proposed which estimates the DAG topology and the corresponding representation in the domain of hierarchical class selection variables jointly, based on multiple data types, i.e., SK/DK data as well as GI-profile data. The GI-GENIE problem depicted in (3.23) is an ILP and can be also solved efficiently by BB and BC methods embedded in CPLEX [Flo95], Gurobi [Don] and Mosek [BV11], respectively. The entire DAG topology recovery procedure based on the GI-GENIE algorithm is compactly described by Algorithm 5.

Due to the combinatorial nature of the proposed ILP formulations for graph learning, i.e., the GENIE and the GI-GENIE algorithm, the SEQSCA method has been proposed in Section 3.3. The SEQSCA method utilizes a DAG learning algorithm in order to learn the reliability matrix of a large-scale set of genes in a “divide-and-conquer” fashion.

Finally, in Section 3.4 the performance of the proposed ILP algorithms is evaluated first with respect to synthetic data and secondly with respect to real data. In Subsection 3.4.1, the GENIE and GI-GENIE algorithms are compared to the AIS method of [BJW⁺10] in terms of DAG learning performance in terms of P_{ed} and P_{mis} as defined in Eq. (3.38) and Eq. (3.39), respectively. It has been shown that the ILP based DAG learning algorithms are superior to the AIS method of [BJW⁺10] that is structurally a heuristic method without any guarantees to yield the global optimum. In Subsection 3.4.2, the proposed ILPs as well as the AIS method of [BJW⁺10] are separately used in combination with the SEQSCA algorithm in order to compute the corresponding reliability matrices of the large set of genes under study with real data. The reliability matrices of the considered methods reflect GI-networks of simplified genetic

interactions compared to the ones of [BJW⁺10] and have been evaluated according to general biological assumptions and the yeast database of [yea]. It has been shown again that the ILP based graph learning formulations are superior to the heuristic AIS method of [BJW⁺10].

Chapter 4

Machine Learning Algorithms for Graph Learning

As illustrated in Chapter 3, the proposed ILP-algorithms, i.e., the GENIE and the GI-GENIE algorithm, are based on solving constraint discrete optimization problems. In general, both problems are NP-hard which implies a poor scalability with respect to the number of genes under study. In this chapter, well known ML-algorithms for classification are utilized to learn network/DAG topologies according to [BJW⁺10] as presented in Section 2.2 approximately, providing a tradeoff between topology estimation quality and scalability.

In particular, the major contributions of this chapter are summarized in the following:

- In Section 4.1, two different feature attribute designs, that are based on SK/DK and GI-profile data, are proposed. The first design approach aims at constructing feature attributes that are suitable for learning the hierarchical relationship classes of [BJW⁺10] described in Subsection 2.2.2 where the actual DAG topology is determined according to Algorithm 4 and the learned relationship classes. In contrast to the first design approach, the goal of the second feature attribute design is to construct feature attributes that are suitable for directly learning the GI-network/DAG topology.
- In Section 4.2, Section 4.3 and Section 4.4, the considered ML-methods for learning the hierarchical relationship classes of [BJW⁺10] and the network topology directly are briefly reviewed. In particular, in Section 4.2 the SVM classifier for multi-label classification is described [HL02]. Furthermore, in Section 4.3 the well known random forest (RF) classifier is presented [Bre01]. Finally, in Section 4.4 the currently very popular concept of artificial neural networks (ANN) is briefly explained [Bis95], [GBC16].
- In Section 4.5, the hierarchical relationship class / network topology learning performance of the considered ML-algorithms is evaluated by means of MC- simulations with synthetic data. In essence, the hierarchical relationship class learning performance is evaluated in terms of class estimation accuracy, whereas the ML- algorithms' performance of directly learning the DAG topology is assessed in terms of the normalized number of erroneously estimated edges P_{ed} defined in Eq. (3.38) and the normalized number of missing edges P_{mis} defined in Eq. (3.39).

4.1 Feature Engineering

This section is divided into two parts: first, in Subsection 4.1.1 the feature attribute design for learning DAG topologies in the domain of the hierarchical relationship classes of [BJW⁺10] is illustrated. Secondly, in Subsection 4.1.2 a feature attribute design for learning the DAG topology directly is presented.

4.1.1 Hierarchical Relationship Class Learning

In supervised learning scenarios for classification, feature engineering is always a critical task since there is no right or wrong. In principle, any feature attribute design is valid for any classification task. However, there are designs that work well whereas others are not successful. Hence, to avoid constructing feature attributes that show poor performance in terms of learning accuracy, any reasonable design approach should attempt to aggregate in the feature attributes as much information as possible related to the class labels to be learned. Hence, the feature attributes to learn the hierarchical relationship classes of [BJW⁺10] are designed in such a way to take account for the following design criteria:

1) *per edge feature vector:*

Any considered pair of genes $\{i, j\} \in \mathcal{G} : j > i$ to be classified is associated with one feature vector.

2) *size independence:*

The number of feature attributes, i.e., the length of the feature vector, is independent of the GI-network/DAG size. This ensures that, once a ML-classifier is trained, it can be applied to estimate DAG topologies underlying the SK/DK data of differently sized sets of genes \mathcal{G} under study.

3) *pairwise estimation:*

Given the SK/DK data of a DAG \mathcal{D} with respect to the set of genes \mathcal{G} . Furthermore, let set $A_{\mathcal{D}}$ be given that describes DAG \mathcal{D} in the domain of the hierarchical relationship classes. Then the feature vectors are designed in such a way that an estimate $\hat{A}_{\mathcal{D}}$ of $A_{\mathcal{D}}$ can be computed in a pairwise fashion. More specifically, the interaction class for each pair of genes $\{i, j\} \in \mathcal{G} : j > i$, i.e., each $\alpha_{k,i,j}$, is estimated by one of the considered supervised ML-algorithms for classification only based on the feature vector for that specific pair. Hence, the feature vector design allows for pairwise computation of $\hat{A}_{\mathcal{D}}$.

4) *information integration:*

Any feature vector is composed of two parts: the first part contains the data model of [BJW⁺10], i.e., the mismatch scores of Eq. (3.1) for all $k \in \mathcal{K}$, while the second part captures statistical information about the coupling among the class selection variables $\alpha_{k,i,j}$.

Following the above mentioned coarse criteria, the feature vector of gene pair $\{i, j\} \in \mathcal{G} : j > i$ is denoted as

$$\mathbf{x}(i, j) = [\mathbf{x}_{\text{sc}}^T(i, j), \mathbf{x}_{\text{coup}}^T(i, j)]^T \quad (4.1)$$

where

$$\mathbf{x}_{\text{sc}}(i, j) = \frac{1}{\max\{s_1(i, j), \dots, s_5(i, j)\}} [s_1(i, j), \dots, s_5(i, j)]^T \quad \forall \{i, j\} \in \mathcal{G} : j > i \quad (4.2)$$

reflects the data model of [BJW⁺10] and $\mathbf{x}_{\text{coup}}^T(i, j)$ aggregates information about the logical dependencies among the class selection variables $\alpha_{k,i,j}$. Since the set of class selection variables $A_{\mathcal{D}}$, corresponding to DAG \mathcal{D} , describes DAG \mathcal{D} in a different domain, the class selection variables $\alpha_{k,i,j}$ in $A_{\mathcal{D}}$, which encode hierarchical relationship information about the gene pairs in DAG \mathcal{D} , are coupled with each other by the topology of this particular DAG. Note that $\mathbf{x}_{\text{sc}}(i, j)$ is normalized to the maximum mismatch score $\max\{s_1(i, j), \dots, s_5(i, j)\}$ in order to be of comparable magnitude to the feature attributes contained in $\mathbf{x}_{\text{coup}}^T(i, j)$, since the considered ML-algorithms are known to be sensitive with respect to their classification performance to feature attributes of different value domains. In other words, having feature attributes of vastly differing magnitude ranges can be considered as an implicit weighting of the attributes which is unwanted in most cases. In the following, the construction of the feature attributes contained in $\mathbf{x}_{\text{coup}}^T(i, j)$ is explained. Furthermore, an intuitive interpretation of the feature attributes contained in $\mathbf{x}_{\text{coup}}^T(i, j)$ is presented in order to convey a better understanding about the information encoded in $\mathbf{x}_{\text{coup}}^T(i, j)$.

Let the least-mismatch model (LMM) of gene pair $\{i, j\}$, i.e., $\text{LMM}(i, j)$, be defined according to Eq. (4.3)

$$\text{LMM}(i, j) = \begin{cases} \arg \min_{k \in \mathcal{K}} s_k(i, j) & \text{if } j > i \\ \arg \min_{k \in \mathcal{K}} s_k(j, i) & \text{if } i > j \end{cases} \quad \forall \{i, j\} \in \mathcal{G} \quad (4.3)$$

Furthermore, let

$$\mathcal{L}_k^{(i)} = \{l \in \mathcal{G} \setminus i : \text{LMM}(i, l) = k\} \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{G} \quad (4.4)$$

denote the set of all genes l that are in class k with gene i based on the LMM as defined in Eq. (4.3). Based on Eq. (4.4), the feature attributes $L_k^{(i,j)}$ as defined by Eq. (4.5) below

$$L_k^{(i,j)} = \frac{|\mathcal{L}_k^{(i)} \cap \mathcal{L}_k^{(j)}|}{|\mathcal{L}_k^{(i)} \cup \mathcal{L}_k^{(j)}|} \quad \forall k \in \mathcal{K}, \quad \forall \{i, j\} \in \mathcal{G} : j > i \quad (4.5)$$

are used to construct $\mathbf{x}_{\text{coup}}(i, j)$ according to Eq. (4.6) as described below:

$$\mathbf{x}_{\text{coup}}(i, j) = \left[L_1^{(i,j)}, \dots, L_5^{(i,j)} \right]^T \quad \forall \{i, j\} \in \mathcal{G} : j > i. \quad (4.6)$$

Definition 4.1.1. A path P_{lin} in a DAG \mathcal{D} is a “linear pathway” if all genes/nodes traversed by P_{lin} have at most one incoming edge and at most one outgoing edge.

Definition 4.1.2. Given a DAG \mathcal{D} and set \mathcal{P}_{lin} which contains all linear pathways in DAG \mathcal{D} according to Definition 4.1.1. A longest linear pathway $P_{\text{lin}, \text{max}}$ in DAG \mathcal{D} is a linear pathway in DAG \mathcal{D} that is not contained in any other linear pathway $P_{\text{lin}} \in \mathcal{P}_{\text{lin}}$:

$$\nexists P_{\text{lin}} \in \mathcal{P}_{\text{lin}} : V(P_{\text{lin}, \text{max}}) \subset V(P_{\text{lin}}) \quad (4.7)$$

Based on Definition 4.1.1 and Definition 4.1.2, respectively, the idea beyond designing $\mathbf{x}_{\text{coup}}(i, j)$ according to Eq. (4.6) is illustrated in the following.

Since $L_k^{(i,j)} \in [0, 1]$ for any relevant pair of genes, it follows that $\mathbf{x}_{\text{coup}}(i, j) \in [0, 1]^{5 \times 1}$ as well. Based on the “naive” LMM classification of Eq. (4.3), vector $\mathbf{x}_{\text{coup}}(i, j) = \mathbf{0}^{5 \times 1}$, where $\mathbf{0}^{M \times N}$ denotes the all zero vector of the dimensions $M \times N$, reflects the case when gene i interacts contrarily to all other genes $l \in \mathcal{G} \setminus \{i, j\}$ compared to gene j . On the other hand, vector $\mathbf{x}_{\text{coup}}(i, j) = \mathbf{1}^{5 \times 1}$, where $\mathbf{1}^{M \times N}$ denotes the all one vector of the dimensions $M \times N$, reflects the case when genes i, j interact identically with all other genes $l \in \mathcal{G} \setminus \{i, j\}$ in the network.

Seen from a topological perspective, the more $\mathbf{x}_{\text{coup}}(i, j)$ is approaching $\mathbf{1}^{5 \times 1}$, the more likely it is that both genes i, j are situated in the same longest linear pathway according to Definition 4.1.2 in DAG \mathcal{D} , with $\mathbf{x}_{\text{coup}}(i, j) = \mathbf{1}^{5 \times 1}$ denoting that genes i, j are in the same longest linear pathway with empirical probability 1. On the other hand, the more $\mathbf{x}_{\text{coup}}(i, j)$ is approaching $\mathbf{0}^{5 \times 1}$, the more likely it is that both genes i, j are not situated in the same longest linear pathway according to Definition 4.1.2 in DAG \mathcal{D} , with $\mathbf{x}_{\text{coup}}(i, j) = \mathbf{0}^{5 \times 1}$ denoting that genes i, j are not in the same longest linear

pathway with empirical probability 1. Note that in this case, both genes i, j must not be situated in a linear pathway at all.

In essence, the feature attributes in $\mathbf{x}_{\text{coup}}(i, j)$ capture topology information which is based on the LMM defined in Eq. (4.3). Moreover, the captured topology information in $\mathbf{x}_{\text{coup}}(i, j)$ translates to information about the coupling among the class selection variables $\alpha_{k,i,j}$, since there is a strong relationship between the topology $\mathcal{E}_{\mathcal{D}}$ of a DAG \mathcal{D} and the coupling of the $\alpha_{k,i,j} \in A_{\mathcal{D}}$ and the $\omega_{i,j} \in \Omega(\mathcal{D})$, respectively, as elucidated in Subsection 2.2.3.

For the purpose of training the considered ML-algorithms, a set of training data must be generated. In particular, any feature vector $\mathbf{x}(i, j)$ used for training must be assigned a corresponding class label $y(i, j)$ as defined below in Eq. (4.8):

$$y(i, j) = \begin{cases} 1 & \text{if } \alpha_1(i, j) = 1 \text{ in DAG } \mathcal{D} \\ 2 & \text{if } \alpha_2(i, j) = 1 \text{ in DAG } \mathcal{D} \\ 3 & \text{if } \alpha_3(i, j) = 1 \text{ in DAG } \mathcal{D} \\ 4 & \text{if } \alpha_4(i, j) = 1 \text{ in DAG } \mathcal{D} \\ 5 & \text{if } \alpha_5(i, j) = 1 \text{ in DAG } \mathcal{D} \end{cases} \quad \forall \{i, j\} \in \mathcal{G} : j > i. \quad (4.8)$$

It is important to remark that the SK data is related to the DK data by a DAG \mathcal{D} that describes the hierarchical relationships among the genes under study. Hence, the label of $y(i, j)$ of each feature vector $\mathbf{x}(i, j)$ is determined by DAG \mathcal{D} that is underlying the observed phenotypes of the current knockout experiment.

Finally, with the feature vectors $\mathbf{x}(i, j)$ according to Eq. (4.1) and the class labels $y(i, j)$ according to Eq. (4.8), a training set \mathcal{T} is constructed according to Eq. (4.9)

$$\mathcal{T} = \{\mathbf{x}_t, y_t\}_{t=1}^{T_r} \quad (4.9)$$

that consists of T_r labeled data samples $\{\mathbf{x}_t, y_t\}$ from various knockout experiments where each specific training sample t stands for a specific pair of genes $\{i, j\}$.

4.1.2 Direct DAG Topology Learning

In order to find a suitable feature representation of the SK/DK data and the GI-profile data, respectively, to learn the DAG topology directly with the considered ML-algorithms, it is reasonable again to take account of items 1) to 4) stated in Subsection 4.1.1.

Let

$$\tilde{\mathbf{x}}(i, j) = [\mathbf{x}_{\text{sc}}^T(i, j), \mathbf{x}_{\text{coup}}^T(i, j)]^T \quad (4.10)$$

denote the feature vector of gene pair $\{i, j\} \in \mathcal{G} : j > i$ in the case of direct DAG topology learning, where

$$\mathbf{x}_{\text{sc}}(i, j) = \frac{1}{\max\{s_1(i, j), \dots, s_5(i, j)\}} [s_1(i, j), \dots, s_5(i, j)]^T \quad \forall \{i, j\} \in \mathcal{G} : j > i \quad (4.11)$$

reflects the data model of [BJW⁺10] presented in Subsection 2.2.2. In case of direct topology learning, $\mathbf{x}_{\text{coup}}(i, j)$ only contains the GI-profile data $\rho(i, j)$ which can be interpreted as a measure of similarity of the interactions between gene i and all other genes compared to the interactions between gene j and all other genes. Similarly to Subsection 4.1.1, in order to train the considered ML-algorithms, a set of training data must be generated. Therefore, any feature vector $\mathbf{x}(i, j)$ used for training must be assigned a corresponding class label $y(i, j)$ as defined below in Eq. (4.12):

$$y(i, j) = \begin{cases} 0 & \text{if } \{i, j\}, \{j, i\} \notin \mathcal{E}_{\mathcal{D}} \\ 1 & \text{if } \{i, j\} \in \mathcal{E}_{\mathcal{D}} \\ 2 & \text{if } \{j, i\} \in \mathcal{E}_{\mathcal{D}} \end{cases} \quad \forall \{i, j\} \in \mathcal{G} : j > i. \quad (4.12)$$

where $y(i, j) = 0$ denotes that there is no edge between genes i, j in DAG \mathcal{D} , $y(i, j) = 1$ states that there is an edge from gene i to gene j in DAG \mathcal{D} and finally $y(i, j) = 2$ denotes that there is an edge from gene j to i in DAG \mathcal{D} .

Based on feature vectors $\mathbf{x}(i, j)$ according to Eq. (4.10) and the class labels $y(i, j)$ according to Eq. (4.12), a training set \mathcal{T} is constructed according to Eq. (4.13)

$$\mathcal{T} = \{\mathbf{x}_t, y_t\}_{t=1}^{T_r} \quad (4.13)$$

that consists of T_r labeled data samples $\{\mathbf{x}_t, y_t\}$ from various knockout experiments where each specific training sample t stands for a specific pair of genes $\{i, j\}$.

4.2 Support Vector Machine Approach

In this section, the multi-class SVM that is based on a specifically trained set of binary SVMs is briefly explained.

Given a set of training data \mathcal{T} according to Eq. (4.9)/Eq. (4.13), in order to cope with a target variable y_t that can take more than two possible values, i.e., y_t is multi-categorical,

multiple binary SVMs according to the well known one-versus-one (OVO) design [HL02] are trained. In particular, for each pair of classes $\{k, l\} \in \{1, \dots, N_c\} : l > k$ a binary SVM is trained, i.e., the normal vector $\mathbf{w}^{k,l}$ and the shift $b^{k,l}$ of the corresponding separating hyperplane, that classifies between classes k and l , respectively, is computed. Hence, $\frac{(N_c^2 - N_c)}{2}$ binary SVMs are trained. According to [HL02], the optimization problem that has to be solved in order to train an SVM that classifies between classes k and l is given by

$$\min_{\mathbf{w}^{k,l}, b^{k,l}, \zeta^{k,l}} \quad \frac{1}{2}(\mathbf{w}^{k,l})^T \mathbf{w}^{k,l} + C \sum_t \zeta_t^{k,l} \quad (4.14a)$$

$$\text{s. t. } (\mathbf{w}^{k,l})^T \phi(\mathbf{x}_t) + b^{k,l} \geq 1 - \zeta_t^{k,l} \text{ if } y_t = k \quad (4.14b)$$

$$(\mathbf{w}^{k,l})^T \phi(\mathbf{x}_t) + b^{k,l} \leq \zeta_t^{k,l} - 1 \text{ if } y_t = l \quad (4.14c)$$

$$\zeta_t^{k,l} \geq 0 \quad (4.14d)$$

$$\mathbf{w}^{k,l} \in \mathbb{R}^{L \times 1}, b^{k,l} \in \mathbb{R}, \zeta_t^{k,l} \in \mathbb{R} \quad (4.14e)$$

where $\mathbf{w}^{k,l}$, $b^{k,l}$ describe the normal vector and the shift from the origin, respectively, of the separating hyperplane that classifies between classes k and l . The nonlinear mapping function $\phi(\cdot)$ transforms the original feature vectors \mathbf{x}_t into a high dimensional feature space in order to obtain a data representation $\phi(\mathbf{x}_t) \forall t$ that allows for a better separation in the class labels y_t . The auxiliary variables $\zeta_t^{k,l}$ allow for misclassification if the data is not separable. For computational efficiency, problem (4.14a) is usually solved in its dual domain using the kernel trick where inner products of the nonlinear mapping $\phi(\cdot)$ of the data is replaced by a proper kernel function/kernel. Common choices of the kernel function are:

- Polynomial function: $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^{d_r}$ with degree $d_r > 0$
- Exponential/Gaussian radial basis function (RBF): $K(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x}^T \mathbf{x}'\|^2}$ with parameter γ

Note that the case of no kernel function in use is commonly referred to as “linear kernel”. This convention is also adopted throughout this thesis. Given a set of $\frac{(N_c^2 - N_c)}{2}$ trained binary SVMs, the class label $y \in \{1, \dots, N_c\}$ for an unobserved feature vector \mathbf{x} is predicted according to the well known “max-wins” strategy [HL02].

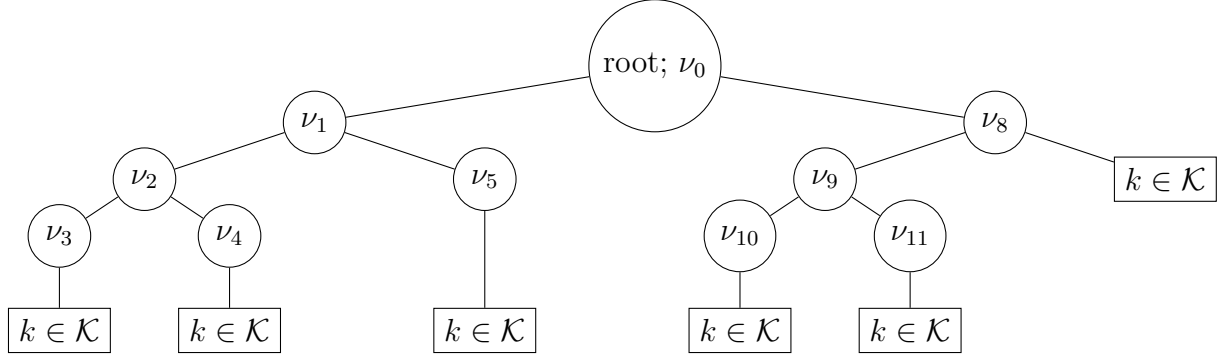


Figure 4.1. General illustration of a binary decision tree T according to Eq. (4.15). The nodes v_1, \dots, v_{11} are associated with their corresponding five-tuple that describe the splits at the respective nodes. Nodes $v_3, v_4, v_5, v_8, v_{10}, v_{11}$ are succeeded by “rectangular”-shaped terminal nodes that select the class of highest probability as the label y for a given feature vector \mathbf{x} .

4.3 Random Forest Classifier Approach

In this section, the Random Forest Classifier [Bre01] for multi-categorical target variables/labels y is briefly revised. First the concept of binary Decision Trees (DTs) [SL91] for multiclass-classification is briefly presented which is afterwards extended to yield the RFC.

A binary decision tree T can be described by the following recursive representation

$$T = \{\nu_v = (v, m, \psi, v^L, v^R), T^L, T^R\} \quad (4.15)$$

starting from the root node where the five-tuple $\nu_v = (v, m, \psi, v^L, v^R)$ describes a binary partition of the entire feature space and T^L and T^R denote the subtrees defined on the left and the right sets of the partition, respectively [SL91]. The depth d of a binary DT is the maximal number of splits that have been traversed in order to get from the root node to a leaf node. In Figure 4.1, a binary DT example is illustrated.

Starting from the root node, a binary decision tree sequentially partitions the entire feature space \mathbb{R}^M into subspaces such that the resulting subspaces are increasingly likely to be governed by only one class of the categorical target variable $y \in \{1, \dots, N_c\}$, where N_c denotes the number of classes. In order to perform the above mentioned feature space partitioning, starting at the root node the entire feature space, i.e., \mathbb{R}^M , is split at each subsequent internal node v by one feature attribute m according to a threshold value ψ that divides the current space at node v into a *left* v^L and a *right*

v^R feature subspace. Particularly, the left feature subspace v^L at v is obtained by splitting the entire feature space of node v at the m -th feature attribute, denoted by $(\mathbf{x})_m$, according to $(\mathbf{x})_m < \psi$. Hence, all elements of the feature space at node v , for which $(\mathbf{x})_m < \psi$ holds, belong to v^L . Similarly, all elements of the feature space at node v , for which $(\mathbf{x})_m \geq \psi$ holds, belong to v^R . Ideally, each leaf node in a DT has a feature subspace that corresponds to one class only, i.e., during training the subspace was completely governed by training samples of one class. However, in practice it is usually sufficient that the feature subspace corresponding to a leaf node is mainly governed by one class. Consequently, classification is done at the leaf nodes according to the most dominant class in the leaf node's feature subspace.

In order to train a DT, i.e., to learn the tree structure and the splits (feature attribute m and the corresponding ψ) at the nodes v , based on a set of training data $\mathcal{T} = \{\mathbf{x}_t, y_t\}_{t=1}^{T_r}$, there are many well known algorithms as, for instance, the CART-algorithm [Cra89], the IDE3-algorithm [Qui86] and the C4.5-algorithm [Sal94].

Given an unobserved feature vector $\mathbf{x} \in \mathbb{R}^{M \times 1}$ and a trained decision tree T , the corresponding class label $y \in \{1, \dots, N_c\}$ is estimated by evaluating the feature attributes according to the specified sequence of splits that is determined by the DT T .

A random forest F is essentially a collection of B binary decision trees T , i.e., $F = \{T^b\}_{b=1}^B$, that have been trained on a data set \mathcal{T} of size T_r in a randomized fashion. In order to train decision tree T^b , a “bootstrapped” data set \mathcal{T}_b is generated by drawing T_r samples $\{\mathbf{x}_t, y_t\}$ uniformly from \mathcal{T} with replacement. Furthermore, the splits at any node v in tree T^b are only considered on uniformly drawn random subsets $\mathcal{M}_f \subset \{1, \dots, m\}$ of feature attributes. Algorithm 7 summarizes that procedure.

Algorithm 7 Random Forest Generation

- 1: **Initialize:**
 training data $\mathcal{T} = \{\mathbf{x}_t, y_t\}_{t=1}^{T_r}$, $\mathbf{x}_t \in \mathbb{R}^{M \times 1}$, $y_t \in \{1, \dots, N_c\}$,
 number of DTs: B , $b = 1$, $F = \emptyset$
 - 2: **for** $b \leq B$ **do**
 - 3: construct bootstrapped training set \mathcal{T}_b from \mathcal{T}
 - 4: construct T^b using \mathcal{T}_b according to algorithms [Cra89], [Qui86] or [Sal94]
 such that the splits at any node v in tree T^b are only considered on
 uniformly drawn random subsets $\mathcal{M}_f \subset \{1, \dots, m\}$ of feature attributes
 - 5: update: $F = F \cup T^b$
 - 6: update: $b \leftarrow b + 1$
 - return** \implies random forest F
-

Finally, given an unobserved feature vector $\mathbf{x} \in \mathbb{R}^{M \times 1}$ and a trained random forest F , the corresponding class label $y \in \{1, \dots, N_c\}$ can be estimated based on the single

decisions of the trees T^b of the forest F in several ways, where the most common decision rule is the majority decision.

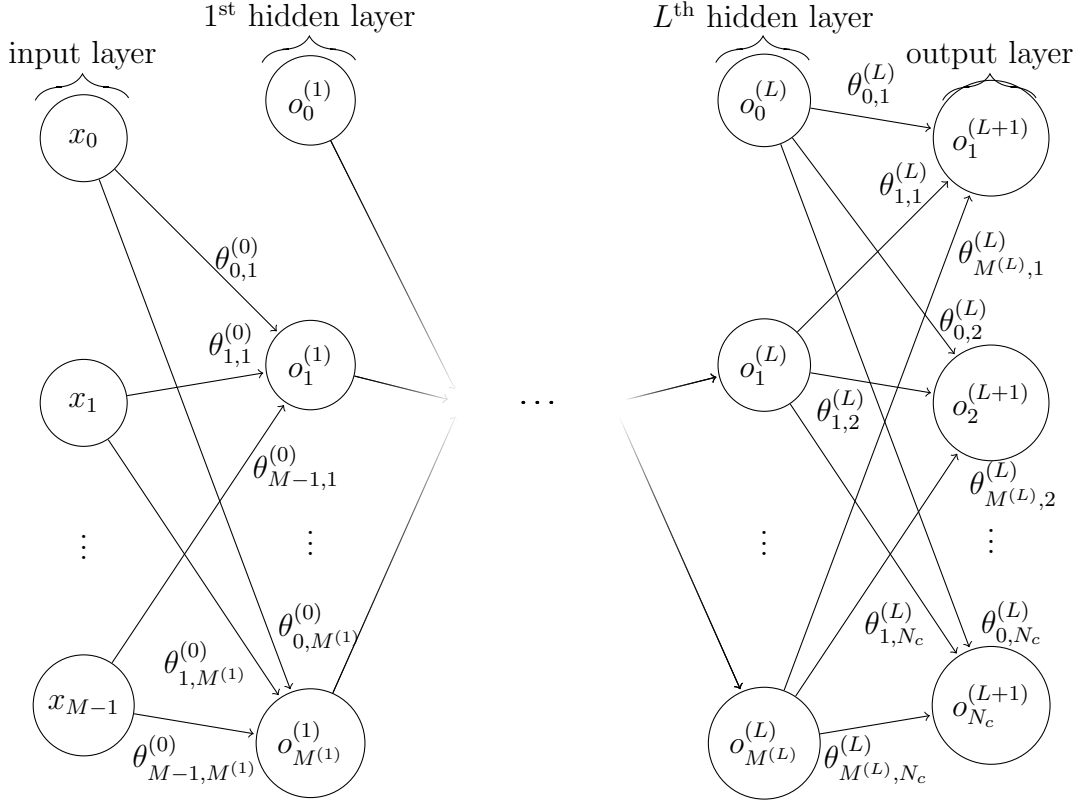


Figure 4.2. Schematic illustration of a non-recurrent artificial neural network (ANN)/ multi-layer perceptron (MLP) with m input nodes, L hidden layers and N_c output nodes. The l -th hidden layer consists of $M^{(l)}$ neurons. The weights θ connect neighbouring layers.

4.4 Artificial Neural Network Approach

In this section, the concept of artificial neural networks [Bis95], [GBC16] is presented and briefly explained. Especially in recent years, there has been increasing attention to ANNs that are widely applied in various fields of science and technology. Inspired by the architecture of animal brains, ANNs are complex mathematical models designed to perform cognitive tasks as for instance, object recognition and scene interpretation in images, speech recognition, text interpretation and text generation. From a mathematical perspective, such cognitive tasks translate into basic mathematical problems that are classification and regression, respectively.

As the name suggests, an ANN is a network of nodes that are connected with each other by weighted edges where the nodes of an ANN are typically arranged in layers. There are different types of layers where each of them is associated with a different functionality. The first layer of an ANN is always denoted as the input layer where the

data is feed into the network. The final layer is denoted as the output layer and yields the quantity of interest, for instance, a regression value or a class label, respectively, assigned to the given input data by the ANN. The layers in between the input and the output layer are denoted as the hidden layers. In general, any network topology, that propagates the input data/signals to the output layer, is possible. However, in practice there have been established specific ANN topologies depending on the field of application. On a coarse scale, ANNs can be divided into two categories that are *non-recurrent* and *recurrent* ANNs. The topology of non-recurrent ANNs do not have cycles whereas recurrent ANNs exhibit cycles in their network topology. For instance, in the field of image processing and image recognition, non-recurrent convolutional artificial neural networks (CNN)s, which have a two dimensional node architecture per layer, are recognized as the best choice [GBC16], [LB⁺95]. On the other hand, in the field of sequence prediction, recurrent ANNs are considered to be the prominent choice [Sch15].

Due to its simplicity, the most common ANN architecture is the non-recurrent feed-forward network. The non-recurrent feedforward architecture contains no cycles and each layer of nodes is only connected with its neighbouring layers. Hence, the network architecture of such ANNs forms a bipartite graph. In this thesis, only non-recurrent ANNs with a network topology, that corresponds to a bipartite graph are, considered.

In general, each node in an ANN mimics a biological neuron and the edges model the signal propagation between the neurons. Similar to its biological counterpart, a neuron in an ANN receives inputs from multiple other neurons where those incoming signals are weighted with their corresponding edge weights. At each neuron, the incoming weighted signals are summed up and given as the argument of a non-linear function commonly referred to as the neural function. The outcomes of the neural functions represent the outgoing signals of the neurons which serve as incoming signals for other neurons. Consequently, by forwarding the input data/signals through the network, the ANN processes its input data in order to yield the desired regression/classification outcome. In the following, the non-recurrent ANN in bipartite graph architecture for the purpose of classification is briefly revisited.

In mathematical terms, an ANN with L hidden layers that classifies a feature vector $\mathbf{x} \in \mathbb{R}^{M \times 1}$ to one specific class label $y \in \{1, \dots, N_c\}$ can be written as a function

$$h_{\boldsymbol{\theta}} : \mathbf{x} \in \mathbb{R}^{M \times 1} \mapsto y \in \{1, \dots, N_c\} \quad (4.16)$$

which is parameterized by the coefficient vector

$$\boldsymbol{\theta} = [\boldsymbol{\theta}^{(0),T}, \dots, \boldsymbol{\theta}^{(L),T}]^T \quad (4.17)$$

where coefficient vector $\boldsymbol{\theta}^{(l),T}$ for $l \in \{0, \dots, L\}$ models the connection between neurons of layer l with neurons of layer $l + 1$. Furthermore, the coefficient vectors $\boldsymbol{\theta}^{(l),T}$ for $l \in \{0, \dots, L\}$ are characterized by

$$\boldsymbol{\theta}^{(l)} = [\boldsymbol{\theta}_1^{(l),T}, \dots, \boldsymbol{\theta}_{M^{(l)}}^{(l),T}]^T \quad (4.18)$$

where

$$\boldsymbol{\theta}_{m'}^{(l)} = [\theta_{m',1}^{(l)}, \dots, \theta_{m',M^{(l+1)}}^{(l)}]^T \quad \text{for } m' \in \{1, \dots, M^{(l)}\} \quad (4.19)$$

models the connection between neuron $m' \in \{1, \dots, M^{(l)}\}$ of layer l and all the neurons of layer $l + 1$. Note that the input layer, i.e., the layer containing the data, and the output layer that provides the classification results are denoted by $l = 0$ and $l = L + 1$, respectively.

The activation value of each neuron in any hidden layer and the output layer, respectively, is given by a weighted superposition of all neural activation values of the previous layer. For instance, the activation value of the first neuron in the first hidden layer of the ANN in Fig. (4.2), i.e., $o_1^{(1)}$, is given by

$$o_1^{(1)} = f_n\left(\Lambda_1^{(1)}\right) = f_n\left(\sum_{m'=0}^{M-1} \theta_{m',1}^{(0)} x_{m'}\right) \quad (4.20)$$

where $f_n(\cdot)$ denotes the neural function that models the “stimulation” of the respective neuron and belongs to the class of sigmoid functions and $\Lambda_1^{(1)}$ denotes the network input of this particular neuron. Typically, $f_n(\cdot)$ is selected as a:

- Logistic function: $f_n(x) = \frac{1}{1+e^{-a}}$ for $x \in \mathbb{R}$
- Rectified linear unit function: $f_n(x) = \max(0, x)$ for $x \in \mathbb{R}$
- Hyperbolic tangent function: $f_n(x) = \tanh(x)$ for $x \in \mathbb{R}$

Given a non-recurrent feedforward ANN with L hidden layers that is parameterized by $\boldsymbol{\theta}$ as illustrated in Fig. 4.2 and a set of T_r training samples $\mathcal{T} = \{\mathbf{x}_t, \mathbf{y}_t\}_{t=1}^{T_r}$, the objective/loss function $J(\boldsymbol{\theta})$ to be minimized while training the ANN is given by Eq. 4.25

$$J(\boldsymbol{\theta}) = \frac{1}{2T_r} \sum_{t=1}^{T_r} \sum_{k=1}^{N_c} \left((h_{\boldsymbol{\theta}}(\mathbf{x}_t))_k - \tilde{y}_{t,k} \right)^2 \quad (4.21)$$

where $(h_{\boldsymbol{\theta}}(\mathbf{x}_t))_k$ denotes the value of the output neuron corresponding to class $k \in \{1, \dots, N_c\}$, i.e., the k -th output neuron, and $\tilde{y}_{t,k} \in \{0, 1\}^{N_c \times 1}$ denotes the expanded class label according to Eq. 4.22

$$\tilde{y}_{t,k} = \begin{cases} 1 & \text{if } y_t = k \\ 0 & \text{otherwise} \end{cases} \quad \forall t \in \{1, \dots, T_r\}, \forall k \in \{1, \dots, N_c\}. \quad (4.22)$$

It is important to remark that there are different choices of the loss function in (4.25) for the training procedure possible. However, the quadratic loss function depicted in Eq. (4.25) is a common and frequently used choice. Hence, the optimization problem to be solved in order to train an ANN with fixed network architecture is given by Eq. (4.23)

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad (4.23a)$$

s. t.

$$\boldsymbol{\theta} \in \mathbb{R}^{N_{\theta} \times 1} \quad (4.23b)$$

where the number of coefficients N_{θ} is dictated by the network architecture. It is important to remark that problem (4.23) is non-convex and non-linear and the global optimum of $J(\boldsymbol{\theta})$ is not guaranteed to be found. Training an ANN, i.e., solving problem (4.23), is generally accomplished by gradient-descent algorithms [BV04]. However, computing the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is generally a difficult problem, due to the complex structure of the loss function with respect to $\boldsymbol{\theta}$ that is dictated by the ANN topology. In order to compute the gradient $\nabla J(\boldsymbol{\theta})$, the backpropagation algorithm [LBH15] has been established as the state-of-the-art which is briefly reviewed in the following.

The ANN loss function in (4.25) can be fragmented into sample-wise components as given by Eq. (4.24)

$$J(\boldsymbol{\theta}) = \frac{1}{T_r} \sum_{t=1}^{T_r} J_t(\boldsymbol{\theta}) \quad (4.24)$$

where

$$J_t(\boldsymbol{\theta}) = \frac{1}{2} \sum_{k=1}^{N_c} \left((h_{\boldsymbol{\theta}}(\mathbf{x}_t))_k - \tilde{y}_{t,k} \right)^2 \quad (4.25)$$

denotes the loss associated with training sample t . The gradient $\frac{dJ}{d\boldsymbol{\theta}}$ can be also decomposed with respect to the training samples as given by Eq. (4.26) as

$$\frac{dJ}{d\boldsymbol{\theta}} = \frac{1}{T_r} \sum_{t=1}^{T_r} \frac{dJ_t}{d\boldsymbol{\theta}}. \quad (4.26)$$

Therefore, the computation of the full gradient $\frac{\partial J}{\partial \theta}$ can be done in a sample-wise fashion which is exploited in the following. In order to compute the derivative $\frac{\partial J_t}{\partial \theta_{m', m''}^{(l)}}$ of weight coefficient $\theta_{m', m''}^{(l)}$, $m' \in \{0, \dots, M^{(l)}\}$ and $m'' \in \{0, \dots, M^{(l+1)}\}$, that connects neuron m' of layer l with neuron m'' of layer $l + 1$, the chain rule of computing the derivative of a composite of two or more functions is applied. In particular, the derivative of J_t with respect to weight $\theta_{m', m''}^{(l)}$ can be decomposed into three partial derivatives as given by Eq. (4.27)

$$\frac{\partial J_t}{\partial \theta_{m', m''}^{(l)}} = \frac{\partial J_t}{\partial o_{m''}^{(l+1)}} \frac{\partial o_{m''}^{(l+1)}}{\partial \Lambda_{m''}^{(l+1)}} \frac{\partial \Lambda_{m''}^{(l+1)}}{\partial \theta_{m', m''}^{(l)}} \quad (4.27)$$

where $\frac{\partial J_t}{\partial o_{m''}^{(l+1)}}$ denotes the partial derivative with respect to the output $o_{m''}^{(l+1)}$ of neuron m'' in layer $l + 1$, $\frac{\partial o_{m''}^{(l+1)}}{\partial \Lambda_{m''}^{(l+1)}}$ denotes the partial derivative of the output $o_{m''}^{(l+1)}$ of neuron m'' in layer $l + 1$ with respect to $\Lambda_{m''}^{(l+1)}$ that is the input of weighted outputs of all neurons of layer l , and finally, $\frac{\partial \Lambda_{m''}^{(l+1)}}{\partial \theta_{m', m''}^{(l)}}$ denotes the partial derivative of the input of weighted outputs of all neurons of layer l , i.e., the *network input* $\Lambda_{m''}^{(l+1)}$ of neuron m'' in layer $l + 1$, with respect to the weight $\theta_{m', m''}^{(l)}$ that connects neuron m' of layer l with neuron m'' of layer $l + 1$. In particular, the partial derivative $\frac{\partial \Lambda_{m''}^{(l+1)}}{\partial \theta_{m', m''}^{(l)}}$ is computed as depicted in Eq. (4.28)

$$\frac{\partial \Lambda_{m''}^{(l+1)}}{\partial \theta_{m', m''}^{(l)}} = \frac{\partial}{\partial \theta_{m', m''}^{(l)}} \left(\sum_{\check{m}=0}^{M^{(l)}-1} \theta_{\check{m}, m''}^{(l)} o_{\check{m}}^{(l)} \right) = o_{m'}^{(l)} \quad (4.28)$$

which states that the change in the network input of neuron m'' of layer $l + 1$, i.e., $\Lambda_{m''}^{(l+1)}$, with respect to weight $\theta_{m', m''}^{(l)}$, is given by the output of neuron m' of layer l .

Furthermore, the partial derivative $\frac{\partial o_{m''}^{(l+1)}}{\partial \Lambda_{m''}^{(l+1)}}$ is given by Eq. (4.29)

$$\frac{\partial o_{m''}^{(l+1)}}{\partial \Lambda_{m''}^{(l+1)}} = \frac{\partial}{\partial \Lambda_{m''}^{(l+1)}} f_n(\Lambda_{m''}^{(l+1)}). \quad (4.29)$$

Eq. (4.29) states that the change in the output of neuron m'' of layer $l + 1$ with respect to its network input, i.e., $\Lambda_{m''}^{(l+1)}$, is given by the partial derivative of the neural function in use for the ANN under study. Note that the neural function $f_n(\cdot)$ is chosen in such a way that the derivative or subderivative can be always computed. In order to compute the partial derivative $\frac{\partial J_t}{\partial \theta_{m', m''}^{(l)}}$ two cases have to be considered:

case 1: neuron m'' , which is connected via weight $\theta_{m',m''}^{(l)}$ with neuron m' of the previous layer, is a node in the output layer of the ANN, i.e., an output neuron

case 2: neuron m'' , which is connected via weight $\theta_{m',m''}^{(l)}$ with neuron m' of the previous layer, is a node in a hidden layer of the ANN, i.e., an inner neuron.

In the first case, i.e., neuron m'' of layer $l + 1$ is an output neuron of the ANN, the partial derivative of J_t with respect to the output $o_{m''}^{(l+1)}$ of neuron m'' in the output layer, with layer $l + 1 = L + 1$, is given by Eq. (4.30)

$$\frac{\partial J_t}{\partial o_{m''}^{(l+1)}} = \frac{\partial J_t}{\partial o_k^{(L+1)}} = \frac{\partial J_t}{\partial (h_{\theta}(\mathbf{x}_t))_k} = (h_{\theta}(\mathbf{x}_t))_k - \tilde{y}_{t,k} \quad (4.30)$$

which states that the change in the loss function J_t with respect to the neural output $o_{m''}^{(l+1)}$ is given by the difference between the output of the respective neuron, i.e., $o_{m''}^{(l+1)} \equiv (h_{\theta}(\mathbf{x}_t))_k$ with $m'' = k$, and the corresponding label $\tilde{y}_{t,k}$ provided by the training data.

In the second case, i.e., neuron m'' of layer $l + 1$ is situated in a hidden layer of the ANN, the computation of $\frac{\partial J_t}{\partial o_{m''}^{(l+1)}}$ is more difficult. However, as shown in the following, the partial derivative $\frac{\partial J_t}{\partial o_{m''}^{(l+1)}}$ can be computed in a recursive fashion which is the fundamental idea beyond the Backpropagation algorithm. Since the output $o_{m''}^{(l+1)}$ of neuron m'' in layer $l + 1$ is passed forward as an input to the neurons of the next hidden layer, the output $o_{m''}^{(l+1)}$ affects the loss function J_t via the neurons in layer $l + 2$.

Furthermore, since the network inputs $\Lambda_{\check{m}}^{(l+2)}$ of each neuron \check{m} of layer $l + 2$ is a function of the output $o_{m''}^{(l+1)}$ of neuron m'' of layer $l + 1$, the loss function J_t can be interpreted as a function of the network inputs $\Lambda_{\check{m}}^{(l+2)}$ as depicted in Eq. (4.31)

$$\frac{\partial J_t(o_{m''}^{(l+1)})}{\partial o_{m''}^{(l+1)}} = \frac{\partial J_t(\Lambda_1^{(l+2)}, \dots, \Lambda_{M^{(l+2)}}^{(l+2)})}{\partial o_{m''}^{(l+1)}} \quad (4.31)$$

Since $o_{m''}^{(l+1)}$ affects the loss function J_t via each neuron in layer $l + 2$, the partial derivatives of J_t of each node in layer $l + 2$ with respect to their network inputs, i.e., $\Lambda_{\check{m}}^{(l+2)}$, have to be summed up in order to quantify the total influence of a change in $o_{m''}^{(l+1)}$ for the loss function J_t . Hence, the total derivative is computed based on the sum of all partial derivatives as described in Eq. (4.32)

$$\frac{\partial J_t(o_{m''}^{(l+1)})}{\partial o_{m''}^{(l+1)}} = \sum_{\check{m}=0}^{M^{(l+2)}} \left(\frac{\partial J_t}{\partial \Lambda_{\check{m}}^{(l+2)}} \frac{\partial \Lambda_{\check{m}}^{(l+2)}}{\partial o_{m''}^{(l+1)}} \right) \quad (4.32)$$

where the RHS of Eq. (4.31) has been obtained by using the chain rule of derivatives. Since $o_{m''}^{(l+2)}$ is a function of the network input $\Lambda_{\check{m}}^{(l+2)}$, as described in Eq. (4.20), the chain rule is applied again in order to substitute $\frac{\partial J_t}{\partial \Lambda_{\check{m}}^{(l+2)}}$ by $\frac{\partial J_t}{\partial o_{\check{m}}^{(l+2)}} \frac{\partial o_{\check{m}}^{(l+2)}}{\partial \Lambda_{\check{m}}^{(l+2)}}$ which allows to re-write the partial derivative $\frac{\partial J_t(o_{m''}^{(l+1)})}{\partial o_{m''}^{(l+1)}}$ in the following form as depicted by Eq. (4.33)

$$\frac{\partial J_t(o_{m''}^{(l+1)})}{\partial o_{m''}^{(l+1)}} = \sum_{\check{m}=0}^{M^{(l+2)}} \left(\frac{\partial J_t}{\partial o_{\check{m}}^{(l+2)}} \frac{\partial o_{\check{m}}^{(l+2)}}{\partial \Lambda_{\check{m}}^{(l+2)}} \frac{\partial \Lambda_{\check{m}}^{(l+2)}}{\partial o_{m''}^{(l+1)}} \right) = \sum_{\check{m}=0}^{M^{(l+2)}} \left(\frac{\partial J_t}{\partial o_{\check{m}}^{(l+2)}} \frac{\partial o_{\check{m}}^{(l+2)}}{\partial \Lambda_{\check{m}}^{(l+2)}} \theta_{m'', \check{m}}^{(l+1)} \right). \quad (4.33)$$

Since the summands of the RHS of Eq. (4.33) are the partial derivatives of the loss function J_t with respect to all weights $\theta_{m'', \check{m}}^{(l)}$, that connect the neurons of layer $l+1$ with the neurons of layer $l+2$, the partial derivative $\frac{\partial J_t}{\partial \theta_{m', m''}^{(l)}}$ can be computed in a recursive fashion given that all partial derivatives $\frac{\partial J_t}{\partial \theta_{m'', \check{m}}^{(l+1)}}$ of the next layer are known. The above described recursion is the essential idea beyond the Backpropagation algorithm. The partial derivatives $\frac{\partial J_t}{\partial \theta_{m', m''}^{(l)}}$ of all weights are computed in a top-down fashion, i.e., starting with the weights that connect the output layer of the ANN with the last hidden layer and finishing with the weights that connect the first hidden layer with the input layer.

In order to account for case 1 and case 2 as described above, the recursive computation of $\frac{\partial J_t}{\partial \theta_{m', m''}^{(l)}}$ is summarized by Eq. (4.34)

$$\frac{\partial J_t}{\partial \theta_{m', m''}^{(l)}} = \delta_{m''}^{(l+1)} o_{m''}^{(l)} \quad (4.34)$$

with

$$\delta_{m''}^{(l+1)} = \frac{\partial J_t}{\partial o_{m''}^{(l+1)}} \frac{\partial o_{m''}^{(l+1)}}{\partial \Lambda_{m''}^{(l+1)}} = \begin{cases} \left((h_{\theta}(\mathbf{x}_t))_k - \tilde{y}_{t,k} \right) \frac{\partial}{\partial \Lambda_{m''}^{(l+1)}} f_n(\Lambda_{m''}^{(l+1)}) & \text{output layer} \\ \left(\sum_{\check{m}=0}^{M^{(l+2)}} \delta_{\check{m}}^{(l+2)} \theta_{m'', \check{m}}^{(l+1)} \right) \frac{\partial}{\partial \Lambda_{m''}^{(l+1)}} f_n(\Lambda_{m''}^{(l+1)}) & \text{hidden layer} \end{cases} \quad (4.35)$$

In practice, the gradient computation via the Backpropagation algorithm involves four steps:

- *initialization:*

All weights aggregated in θ are initialized.

- *feed-forward:*

The training feature vectors \mathbf{x}_t are forwarded through the ANN and the loss J_t for each training example $\{\mathbf{x}_t, y_t\}$ is computed. Furthermore, the outputs of all neurons are stored.

- *sample-wise gradient backpropagation:*

According to the recursive computation rule depicted by Eqs. (4.34) and (4.35), the gradient $\frac{dJ_t}{d\theta}$ is computed.

- *final gradient:*

The final gradient is obtained as the sample-wise gradients according to Eq. (4.26).

Once the gradient has been computed by the Backpropagation algorithm according to the steps described above, the computed gradient can be used by gradient descent methods in order to train the ANN with the given set of training data. After the training phase, class label predictions for unseen feature vectors \mathbf{x} are made by forwarding the data through the ANN and assigning the class label associated with the highest output neuron to the data \mathbf{x} . Finally, it is important to remark that the field of ANNs is ongoing research and many important questions are not yet answered, as for instance, what ANN topology to choose for a given task. Furthermore, training ANNs with many hidden layers denoted as deep networks is non-trivial and causes many problems, as for instance, the vanishing gradient problem [Hoc98], [LBH15].

4.5 Simulation Results

In this section, simulation results on synthetic data are presented in order to evaluate the graph learning performance of the ML approaches presented in Section 4.2, Section 4.3 and Section 4.4, respectively. In particular, in Subsection 4.5.1 the aforementioned ML algorithms are applied to the hierarchical relationship class problem of (3.5) that is to learn the set of hierarchical relationship classes which best describes the observed SK/DK phenotypes. The ML algorithms presented in Section 4.2 to Section 4.4 yield approximate solutions to that problem that provide a good tradeoff between estimation accuracy and scalability with respect to the number of genes under study. In Subsection 4.5.2, the considered ML algorithms are utilized in order to directly learn the DAG topology underlying the observed phenotypes.

4.5.1 Hierarchical Relationship Class Learning

In this subsection, the considered ML algorithms are compared to the GENIE algorithm with respect to their hierarchical relationship class learning performance in terms of the accuracy score that measures the number of correctly estimated classes normalized to the total number of class selection decisions to be made.

The ML algorithms are trained based on training set \mathcal{T} . In particular, the set of training data $\mathcal{T} = \{\mathbf{x}_t, y_t\}_{t=1}^{T_r}$ has been generated according to Eqs. (4.2) and (4.6) and Eq. (4.8), respectively, where the feature vectors \mathbf{x}_t have been computed based on synthetic SK/DK data that has been generated according to steps 1 to 5 of the procedure described in Subsection 3.4.1. The training set \mathcal{T} aggregates the data of 10^4 equally sized DAGs of different data quality regimes, i.e., the SNRs are different in general. Ideally, the data in the training set \mathcal{T} should reflect all possible topology configurations, which DAGs can exhibit, under all possible SNR scenarios, in order to have observed all patterns in the feature attributes that belong to the respective hierarchical relationship classes of [BJW⁺10]. Accounting for this aspect and practical limitations in computational resources, the size of the DAGs is fixed to 10 genes which yields a total number of $T_r = 45 \times 10^4$ training samples $\{\mathbf{x}_t, y_t\}$ which has established as a reasonable choice.

In order to evaluate the performance of the algorithms in terms of the accuracy score with respect to the estimated hierarchical relationship classes of [BJW⁺10], two types of MC simulations are conducted. The number of Monte Carlo runs per SNR and network size sample, respectively, have been set to 200. This choice represents a good tradeoff in terms of computational cost and variance reduction of the performance results shown. In particular, the variance of the different performance results per SNR/network size samples is not sufficiently decreased with higher numbers of MC-runs such that the resulting computational cost is compensated. In the first type of MC simulations, the accuracy score of estimated hierarchical relationship classes is evaluated versus varying SNR values with 200 MC simulations per SNR sample and a fixed size of considered DAGs of 10 genes. In the second type of MC simulations, the accuracy score of estimated hierarchical relationship classes is evaluated versus a changing DAG size with a fixed SNR value of 30dB. Furthermore, 200 MC simulations are conducted per network size sample. The simulation settings for both types of described MC simulations are summarized in Tables 4.5.1 and 4.5.1, respectively.

Each of the considered ML classifiers, i.e., the SVM, the RF as well as the ANN, requires choosing a set of configuration parameters that is obtained by a cross-validation

procedure. In particular, the three considered ML methods for classification have been evaluated on a set of cross-validation test data in terms of the accuracy score, in order to obtain the best parameter configuration for the respective method. Note that the space of configuration parameters of any of the considered ML algorithms for classification is infinite, due to continuous parameter values and parameters with an unlimited degree of freedom, respectively, as for instance, hidden layers in an ANN, number of neurons in an ANN, depth of an RFC, number of DTs in a RFC. Hence, it is common to manually define the parameter set of interest in order to enable a grid-search over the different parameter configurations. The possible parameter configurations of the respective ML classifier methods, considered in this work, are displayed in Table 4.1 in the sense that the possible manually defined choices of the different parameters are given. Hence, the full parameter space of a particular method is given by the Cartesian product of all possible parameter choices of the respective method. It is important to remark that the following parameters of the respective methods

- SVM transformation kernels: *linear, polynomial, RBF*
- RFC maximal depth: $d = 5, d = 7, d = 10$
- ANN neural function: *Logistic function, tanh function, Rectified linear unit function*

are excluded by the cross-validation procedure for the sake of illustrative purposes as shown in the sequel of this section. In particular, C denotes the penalty parameter of the SVM problem formulation depicted in Eq. (4.14) that controls the allowed misclassification in the SVM training procedure. The parameter d_r denotes the degree of the polynomial kernel function and γ denotes a damping factor of the exponential RBF kernel function. In order to translate the original SVM formulation, that deals with two classes only, into the multiclass scenario, there are two common strategies according to [HL02]. The first *decision strategy* is denoted as one-vs-rest, where for each class $k \in \{1, \dots, N_c\}$ a SVM is trained against all the other classes. On the other hand, the one-vs-one decision strategy, which is also adopted in this work, trains a SVM for each pair of classes $\{k, l\} \in \{1, \dots, N_c\} : l > k$ that separates the data corresponding to class k from the data that corresponds to class l . During the training phase of the RFC algorithm, at each node of any binary DT, the feature space is successively split into subspaces with respect to a *split criterion*. The most prominent split criteria are based on the concept of the Information Gain and the Gini-Index [Pal05] which have been also considered in this thesis. In order to obtain a reasonable ANN architecture, the corss-validation procedure has been applied to the proposal network topologies a_n

SVM linear kernel	
C:	$\{0.1, 0.4, 0.5, 0.7, 1, 1.5, 2\}$
decision strategy:	$\{\text{One-vs-One}\}$
SVM polynomial kernel	
C:	$\{0.1, 0.4, 0.5, 0.7, 1, 1.5, 2\}$
decision strategy:	$\{\text{One-vs-One}\}$
poly. degree d_r :	$\{3, 4, 5, 6\}$
SVM RBF kernel	
C:	$\{0.1, 0.4, 0.5, 0.7, 1, 1.5, 2\}$
decision strategy:	$\{\text{One-vs-One}\}$
exp. damping factor γ :	$\{0.5, 1, 2, 2.5, 2.75, 3, 3.25, 5, 7, 8, 9\}$
RFC	
number of trees B :	$\{100, 200, 500\}$
split criterion:	$\{\text{Information gain, Gini-Index}\}$
ANN	
architectures a_n :	$\{(10), (20), (30), (10, 5), (10, 10), (15, 5), (20, 5), (30, 5), (30, 5), (7, 7, 5), (6, 6, 4), (5, 5, 3), (4, 4, 2), (20, 5, 2), (10, 10, 5)\}$

Table 4.1. Set of considered parameter configurations of the cross-validation procedure

as depicted in Table 4.1. In particular, the tuples a_n are of different dimensions where the n -th element of a_n specifies the number of neurons in hidden layer n of the ANN. Consequently, given that $a_n = (4, 4, 2)$ corresponding ANN is composed of three hidden layers where the first hidden layer is composed of 4 neurons, the second hidden layer is composed of 4 neurons and the third hidden layer has 2 neurons.

Based on the parameter configurations of interest depicted in Table 4.1, the cross-validation procedure has yield the following parameterization of the considered methods:

- SVM linear kernel: $\{C : 1, \text{decision strategy: One-vs-One}\}$
- SVM poly. kernel: $\{C : 1, \text{decision strategy: One-vs-One}, d_r : 3\}$
- SVM RBF kernel: $\{C : 1, \text{decision strategy: One-vs-One}, \gamma : 9\}$
- RFC $d = 5$: $\{B : 500, \text{split criterion: Information Gain}\}$
- RFC $d = 7$: $\{B : 500, \text{split criterion: Information Gain}\}$

- RFC $d = 10$: $\{B : 500, \text{split criterion: Information Gain}\}$
- logistic ANN: $\{a_n = (10, 10, 5)\}$
- tanh ANN: $\{a_n = (10, 10, 5)\}$
- ReLu ANN: $\{a_n = (10, 10, 5)\}$

SNR values [dB]:	$\{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$
MC-runs per SNR:	200
s_{\min} :	0
s_{\max} :	1
Network size:	$ \mathcal{G} = 10$

Table 4.2. Simulation setup for ML vs. GENIE “SNR”-MC simulation

Network Size:	$\{10, 15, 20, 25, 30, 40, 50\}$
MC-runs per network size:	200
s_{\min} :	0
s_{\max} :	1
SNR [dB]:	30

Table 4.3. Simulation setup for ML vs. GENIE “Network Size”-MC simulation

The SK/DK data for both types of MC simulations have been generated according to the procedure described by steps 1 to 5 in Subsection 3.4.1 as well. Based on the so generated MC data, the feature vectors \mathbf{x} for both types of MC simulations have been generated according to Eqs. (4.2) and (4.6), respectively. In Figures 4.3 to 4.8, the results of both types of MC simulations of the considered ML algorithms are shown with respect to different parameterizations of the respective ML algorithms. In particular, Figures 4.3 to 4.4 display the estimation performance in terms of the accuracy score against the SNR and the network size, respectively, of multiclass SVMs according to Section 4.2 with three different transformation kernels, i.e., a linear kernel, a polynomial kernel and an exponential kernel denoted as “radial basis function (RBF)”. As shown in Figure 4.3, in a range from 0 dB to 30 dB the considered SVMs perform approximately equally well. However, above a SNR of 30 dB the SVMs using a more complex kernel, i.e., the polynomial kernel and the RBF kernel, start showing a consistently better estimation performance than the SVM utilizing a linear kernel. Especially the SVM using the polynomial kernel outperforms the two other configurations providing a better partitioning of the feature space with respect to the classes. Nevertheless, each of the considered SVMs exhibits a good hierarchical relationship class estimation performance that ranges at 50 dB from 75% accuracy in the case of a

linear kernel to 86% accuracy in the case of a polynomial kernel. As displayed in Figure 4.4, the three different SVM-kernel combinations perform differently well with the network size increasing. Whereas the estimation performance of the multiclass SVM using the RBF kernel stagnates with an increasing network size, the linearly and polynomially kernelized multiclass SVMs show an increase in accuracy with the number of genes in the network increasing. This observation stems from the fact that the feature attributes in the statistical/coupling part of feature vector \mathbf{x} , i.e., in \mathbf{x}_{coup} according to Eq. (4.6), become more expressive in the sense that certain patterns in \mathbf{x}_{coup} reflect a specific topology configuration/class coupling situation with a higher probability. Interestingly, the RBF kernelized multiclass SVM could not capitalize on that fact, hence showing that the RBF kernel in a certain sense over-adapts to the specific data at hand at the cost of a generalization of the data. In essence, the results in Figure 4.4 show that the SVM configurations under study can be trained on small GI-networks/DAGs while being able to estimate the hierarchical relationship classes for much larger DAGs. Hence, in order to predict hierarchical relationship classes for large-scale DAGs it is sufficient to train the SVMs on small DAGs which saves much computational power. Furthermore, for the hierarchical relationship class learning problem, as well as for the given parameter settings and the given feature vector design, the choice of the kernel function substantially affects the estimation performance.

In Figures 4.5 to 4.6, the estimation performance of the RFC algorithm according to Section 4.3 with three different parameter settings is displayed with respect of to the accuracy score against the SNR and the network size, respectively. In particular, each of the three trained RFCs is composed of $B = 500$ binary decision trees where the DTs of the three different random forests are of depths $d = 5$, $d = 7$ and $d = 10$, respectively. As shown in Figure 4.5, in a range from 0 dB to 30 dB the considered RFCs perform approximately equally well. However, above a SNR of 30 dB the RFCs using deeper DTs, i.e., DTs with depth $d = 7$ and depth $d = 10$, start showing a consistently better estimation performance than the RFC built of DTs of depth $d = 5$ only. Note that the performance gap between the RFC with DTs of depth $d = 5$ and the RFC with DTs of depth $d = 7$ is consistently larger than the performance gap between the RFC with DTs of depth $d = 7$ and the RFC with DTs of depth $d = 10$, although the change in depth from $d = 5$ to $d = 7$ is smaller than the change from $d = 7$ to $d = 10$. Hence, this indicates that an increase in DT depth beyond $d = 10$ would not considerably increase the estimation performance of a RFC, i.e., the performance of a RFC is close to saturation in terms of DT depth. In essence, each of the considered RFCs performs considerably well in the high SNR regime in terms of estimation accuracy with up to 95% accuracy shown by the RFC with DT depth $d = 10$. As displayed in Figure 4.6, the RFCs of depths $d = 5$, $d = 7$ and $d = 10$, respectively,

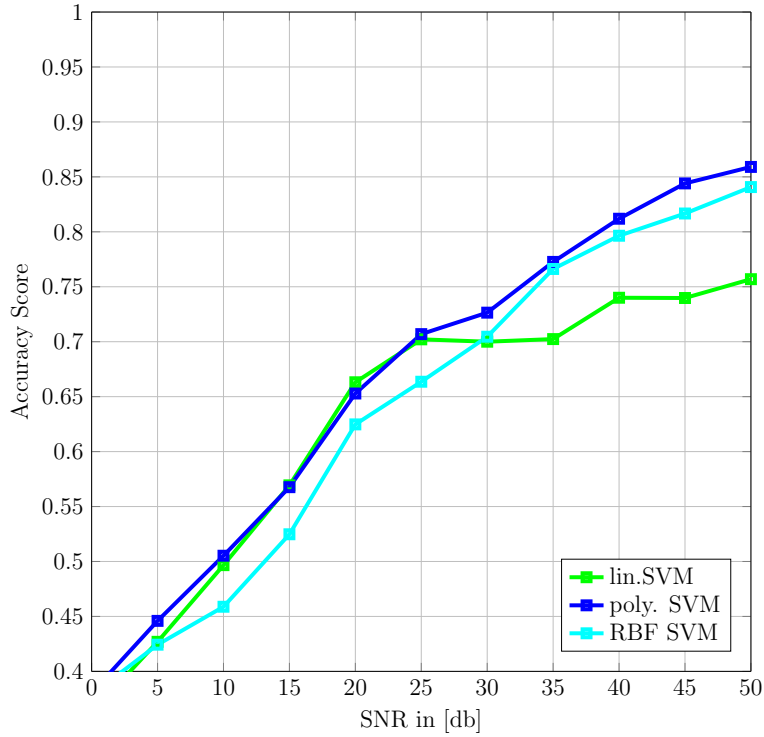


Figure 4.3. Accuracy score vs. SNR; Comparison of different SVM kernels; Polynomial: $d_r : 3$; RBF: $\gamma = 9$

perform approximately equally well with the network size increasing. Again, note that the accuracy of the estimated hierarchical relationship classes is constantly increasing with the networks size which stems from the same fact as explained regarding the SVM results in Figure 4.4. Interestingly, the least complex model, i.e., the RFC with a depth of $d = 5$, performs best. This observation has already been seen in Figure 4.4 where the linear kernel performed best as well. Furthermore, the results in Figure 4.6 show that the RFC configurations under study can be trained on small GI-networks/DAGs while being able to estimate the hierarchical relationship classes for much larger DAGs. Thus, in order to predict hierarchical relationship classes for large-scale DAGs it is sufficient to train the RFCs on small DAGs which saves much computational power.

In Figures 4.7 to 4.8, the estimation performance of the ANN algorithm according to Section 4.4 with three different parameter settings is displayed with respect of to the accuracy score against the SNR and the network size, respectively. In particular, each of the three trained ANNs is composed of 10 input nodes/neurons, a first hidden layer of 10 nodes/neurons, a second hidden layer of 10 nodes/neurons, a third hidden layer of 5 neurons and an output layer of 5 nodes/neurons, however, with different neural functions $f_n(\cdot)$, i.e., the logistic function, the rectified linear unit (Relu) function and the hyperbolic tangent function.

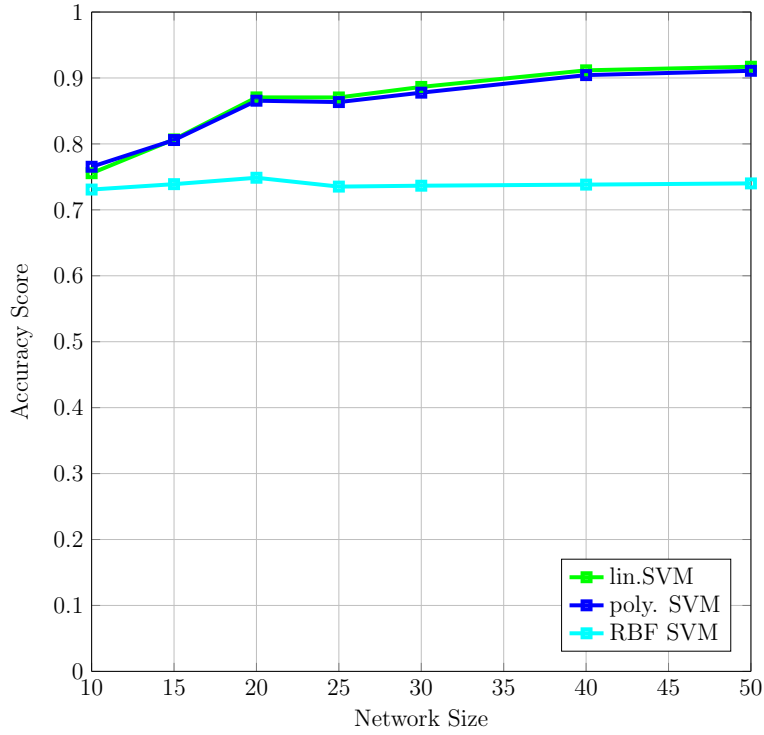


Figure 4.4. Accuracy score vs. network size; Comparison of different SVM kernels; Polynomial: $d_r : 3$; RBF: $\gamma = 9$

Displayed in Figure 4.7, the considered ANNs perform approximately equally well over the SNR range of 0 dB to 25 dB. However, from SNRs above 25 dB the differently parameterized ANNs start exhibit a different performance. While the ANNs using a Relu and hyperbolic tangent function, respectively, still perform comparably well, the ANN utilizing a logistic neural function degrades in performance with respect to the other considered ANN configurations. As displayed in Figure 4.8, the estimation performance of the considered ANNs shows a non-decreasing trend with growing network size which can be explained with the same line of argument as before. In the high SNR regime, the considered ANN classifiers show different performances with the ANN using a Relu function as the neural function performs best. In an overall assessment, for the hierarchical relationship class learning problem, as well as for the given parameter settings and the given feature vector design, the choice of the neural function does affect the estimation performance. Furthermore, it is sufficient again to train the considered ANNs on small GI-networks in order to predict hierarchical relationship classes for large-scale DAGs.

In Figures 4.9 to 4.10, the estimation performance in terms of the accuracy score against the SNR and the network size, respectively, is displayed for the multiclass SVM classifier according to Section 4.2 using a polynomial kernel, the RFC according

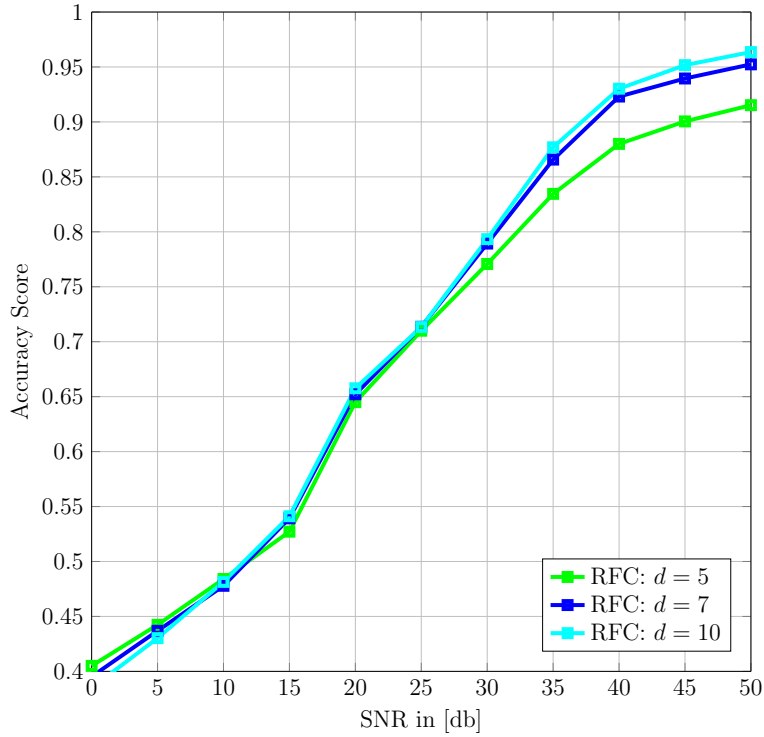


Figure 4.5. Accuracy score vs. SNR; Comparison of RFCs of different DT depths; Number of DTs per forest: $B = 500$

to Section 4.3 of DT depth $d = 10$ and forest size $B = 500$, and the ANN classifier according to Section 4.4 with the same architecture as before (10 input neurons, first hidden layer of 10 neurons, second hidden layer of 10 neurons, third hidden layer of 5 neurons and an output layer of 5 neurons) using the Relu function as the neural function $f_n(\cdot)$. Furthermore, in both figures the above described ML algorithms are compared to the estimation performance of the proposed ILP based algorithms of Section 3.1 and Section 3.2, respectively.

As shown in Figure 4.9, in the low SNR regime the considered ML algorithms perform better than the proposed ILP algorithms showing a more robust hierarchical relationship class learning performance in the low SNR regime. However, above a SNR of 10 dB the GENIE algorithm performs constantly better than the considered ML algorithms exhibiting almost 100% estimation accuracy at a SNR of 50%. Furthermore, in this SNR range the proposed GI-GENIE algorithm clearly outperforms the other methods displaying an excellent estimation performance. In a SNR range from 0 dB to 25 dB, the considered ML algorithms perform approximately equally well. However, above a SNR of 25 dB and the given parameter configurations, the RFC algorithm performs better than the multiclass SVM classifier and the ANN classifier approaching the accuracy performance of the GENIE method. Displayed in Figure 4.10, with an

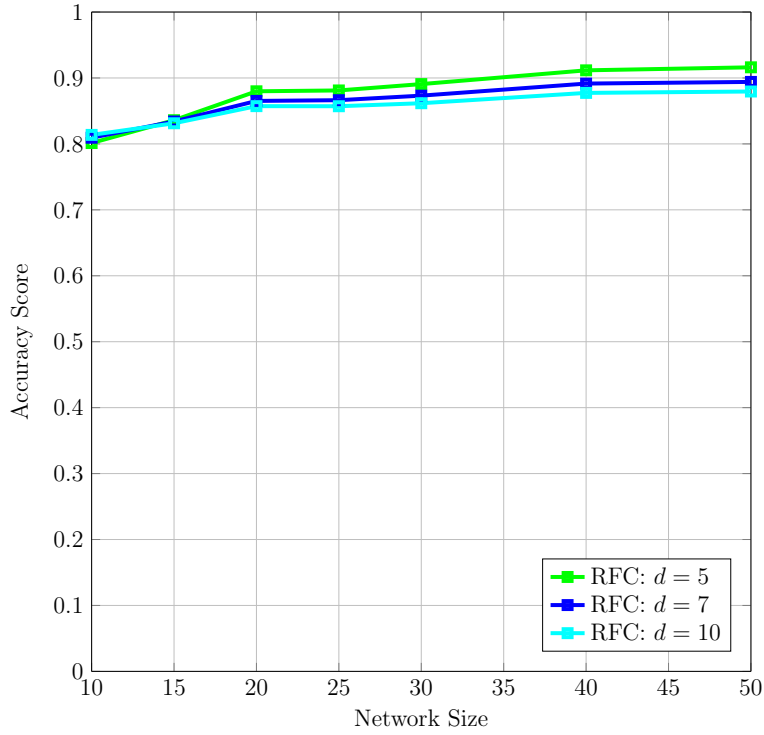


Figure 4.6. Accuracy score vs. network size; Comparison of RFCs of different DT depths; Number of DTs per forest: $B = 500$

increase in GI-network size the learning accuracy increases which can be explained with the same line of argument as before. Not using the feature vectors but the raw SK/DK data instead, the GENIE algorithm shows no such trend which is the same for the GI-GENIE algorithm. Furthermore, the GENIE and the GI-GENIE algorithms are only able to learn the hierarchical relationship class representation of DAGs which have at most 25 and 15 genes, respectively. It is important to remark that above a DAG size of more than 25 genes the GENIE algorithm could not solve one instance of problem (3.5) within the set time limit that was one hour. Moreover, the GI-GENIE algorithm was unable to solve a single instance of (3.23) within this particular time limit for DAGs of more than 15 genes. The ILP based algorithms' inability to find a topology reconstruction for DAGs larger than 15 and 25 genes, respectively, is indicated by an accuracy score of 0%.

This observation reflects the combinatorial nature of the ILP problems in (3.5) and (3.23), respectively, and showcases the weak scalability of this algorithms. In essence, the presented results in Figures 4.9 to 4.10 demonstrate that the considered ML algorithms with the mentioned parameter configurations and the proposed feature vector design in Section 4.1 provide a good tradeoff between learning accuracy and scalability in terms of genes under study.

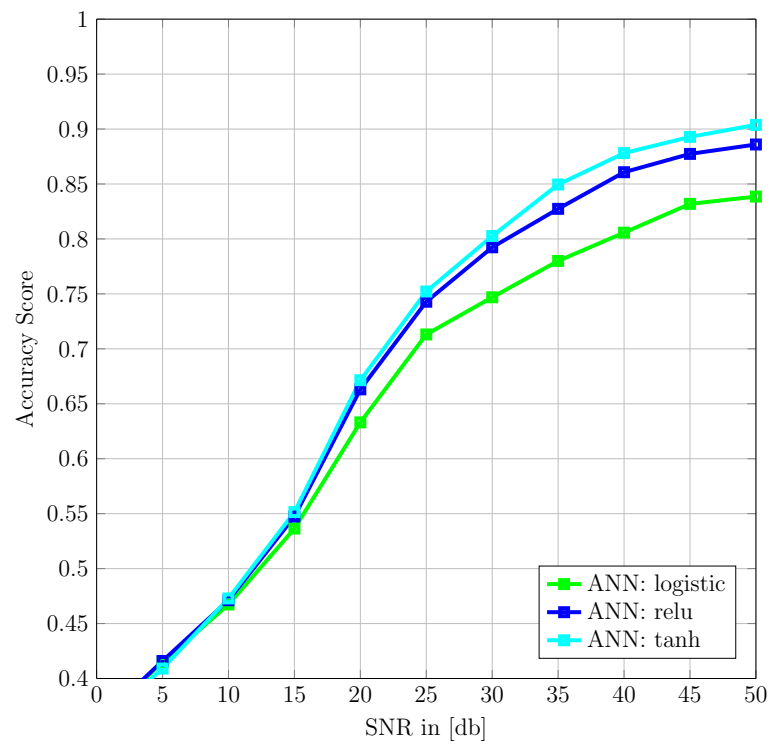


Figure 4.7. Accuracy score vs. SNR; Comparison of ANN classifiers of different neural functions $f_n(\cdot)$

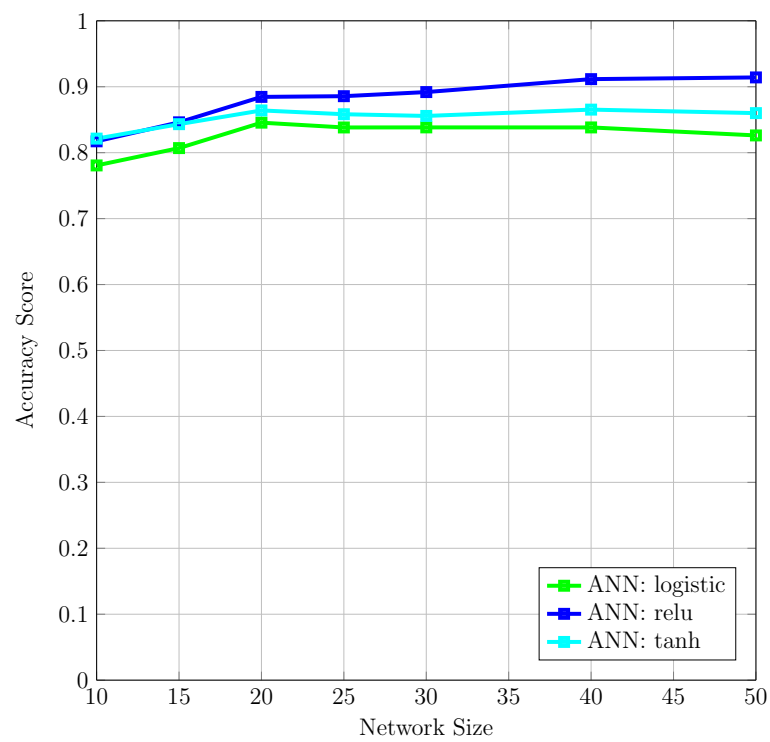


Figure 4.8. Accuracy score vs. network size; Comparison of ANN classifiers of different neural functions $f_n(\cdot)$

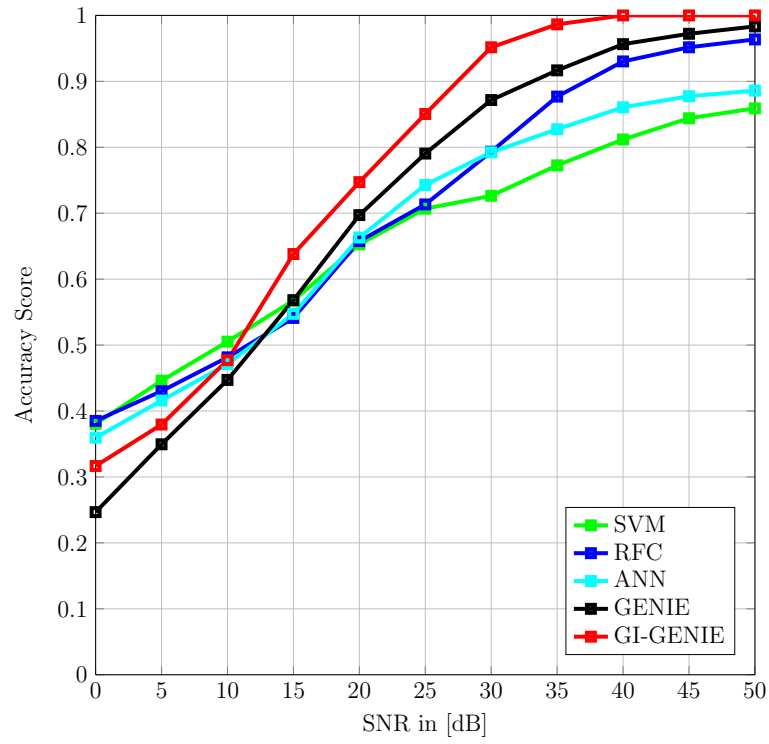


Figure 4.9. Accuracy score vs. SNR; GENIE vs. ML algorithms; SVM: poly. Kernel: $d_r = 3$; RFC: depth $d = 10$, DT number $B = 500$; ANN: Relu neural function; $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.85$

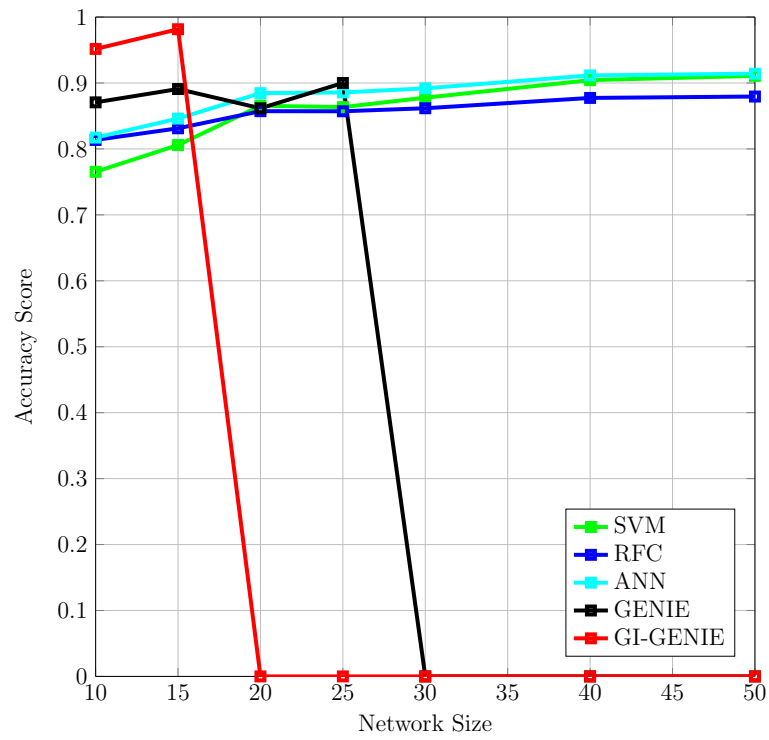


Figure 4.10. Accuracy score vs. network size; GENIE vs. ML algorithms; SVM: poly. Kernel: $d_r = 3$; RFC: depth $d = 10$, DT number $B = 500$; ANN: Relu neural function; $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.85$

4.5.2 Direct DAG Topology Learning

In this subsection, the considered ML algorithms are compared to the GI-GENIE algorithm with respect to their DAG topology learning performance in terms of P_{ed} and P_{mis} according to Eqs. (3.38) and (3.39), respectively.

The ML algorithms are trained based on training set \mathcal{T} . In particular, the set of training data $\mathcal{T} = \{\mathbf{x}_t, y_t\}_{t=1}^{T_r}$ has been generated according to Subsection 4.1.2 where the feature vectors \mathbf{x}_t have been computed based on synthetic SK/DK data that has been generated according to steps 1 to 5 of the procedure described in Subsection 3.4.1. Following the same line of argument of Subsection 4.5.1, the training set \mathcal{T} essentially aggregates the data of 10^4 equally sized DAGs of different data quality regimes, i.e., the SNRs are different in general. The size of the DAGs is fixed to 10 genes which yields a total number of $T_r = 45 \times 10^4$ training samples $\{\mathbf{x}_t, y_t\}$. Following a cross-validation procedure, the parameterization of the considered methods in the case of direct DAG topology learning is identical to the one depicted in Subsection 4.5.1. This observation stems from the fact that the feature vector designs of Subsection 4.1.1 and Subsection 4.1.2, respectively, i.e., the hierarchical relationship class learning design approach and the direct DAG topology learning approach, share the first five feature attributes.

	$y = 0$	$y = 1$	$y = 2$
emp. Pr.	75%	12%	13%

Table 4.4. Empirical probability of class labels y according to Eq. (4.12) to occur in training set \mathcal{T}

In Table 4.4, the empirical distribution of class labels y according to Eq. (4.12) in the training set \mathcal{T} is displayed, where $y = 0$ denotes the case that no edge exists between the considered two genes and $y = 1/y = 2$, respectively, indicate the presence of an edge between the considered two genes with orientation specified according to Eq. (4.12). Note that the predominance of class label $y = 0$ originates from the fact that most DAGs are rather sparse in terms of their number of edges.

In a similar fashion as in Subsection 4.5.1, the performance of the algorithms is evaluated in terms of P_{ed} and P_{mis} , respectively, where two types of MC simulations are conducted, i.e., the considered algorithms are evaluated in terms of P_{ed} and P_{mis} versus a varying SNR and a varying network size, respectively. With the same line of argument as used in Subsection 4.5.1, the number of MC-runs is set to 200 per SNR/network

size sample. In order to compare the GI-GENIE algorithm to the considered ML algorithms, the parameter settings in the case of the “SNR”-MC simulations are identical to the ones in Subsection 4.5.1 whereas the parameter settings in the case of the “Network Size”-MC simulations slightly differ in the considered DAG sizes. For better reference, the parameter configurations of both types of MC simulations are displayed below in Tables 4.5.2 and 4.5.2.

SNR values [dB]:	$\{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$
MC-runs per SNR:	200
s_{\min} :	0
s_{\max} :	1
Network size:	$ \mathcal{G} = 10$

Table 4.5. Simulation setup for ML vs. GI-GENIE “SNR”-MC simulation

Network Size:	$\{6, 7, 8, 9, 10, 12, 15, 20\}$
MC-runs per network size:	200
s_{\min} :	0
s_{\max} :	1
SNR [dB]:	30

Table 4.6. Simulation setup for ML vs. GI-GENIE “Network Size”-MC simulation

In Figures 4.11 to 4.12 and 4.13 to 4.14, the “SNR”-MC simulation and the “Network Size”-MC simulation, respectively, are shown in terms of P_{ed} and P_{mis} with respect to three multiclass SVMs according to Section 4.2 with different transformation kernels, i.e., linear kernel, a polynomial kernel and an exponential kernel denoted as “radial basis function (RBF)”.

In Figure 4.11, the percentage of erroneously detected edges in the sense of P_{ed} according to Eq. (3.38) is displayed. In the low SNR regime, i.e., between a SNR of 0 dB to 20 dB, the performance of the differently kernelized multiclass SVMs is differing considerably. While the multiclass SVM using a polynomial kernel shows a solid performance in terms of P_{ed} , the two other kernelized multiclass SVMs perform worse. In the high SNR regime, i.e., upwards a SNR of 20 dB, the tide is turning with the polynomially kernelized multiclass SVM now exhibiting the worst performance. However, in this SNR region the performance gap is not substantial and the performance of the considered multiclass SVMs is in a low error range of 7% to 14% with respect to P_{ed} . In Figure 4.12, the percentage of missing edges in the sense of P_{mis} according to Eq. (3.39) is displayed versus the SNR. Although the error P_{mis} is constantly decreasing for all of the considered multiclass SVMs, the performance gap between the

considered methods is enormous, especially in the high SNR region. In particular, P_{mis} is constantly decreasing for the polynomially kernelized multiclass SVM reaching an error rate in terms of P_{mis} of 40% at SNR 50 dB. However, the two other multiclass SVM configurations perform considerably better showing a low error rate in terms of P_{mis} of 15% and 8%, respectively.

Turning towards Figure 4.13, the performance of the considered multiclass SVM configurations in terms of P_{ed} against the network size is displayed at 30 dB SNR. Over the entire range of considered network sizes, the considered multiclass SVMs show comparable error rates P_{ed} , especially for networks of more than 12 genes. For networks consisting of 20 genes, the error rates P_{ed} of the considered methods are quite low and situated between 12% and 8% P_{ed} . Displayed in Figure 4.14, the error in terms of missing edges according to P_{mis} against the network size considerably differs among the different multiclass SVMs. The multiclass SVMs using a linear and RBF kernel, respectively, show a performance gap of roughly 10% P_{mis} over the entire range of network sizes. On the contrary, the multiclass SVM using a polynomial kernel exhibits a high error rate P_{mis} of 47% at a network size of 6 genes which increases up to an error of almost 60% P_{mis} for networks with 20 genes. Thus, the performance gap between the multiclass SVM using a polynomial kernel and the other multiclass SVM configurations is enormous.

In essence, the multiclass SVMs using a linear and RBF kernel, respectively, show a strong performance in terms of P_{ed} and P_{mis} with respect to SNR and network size variations. Although trained on smaller networks, the good estimation performance also translates to larger networks of 12 genes and more which saves a lot of computation costs during the training phase. Overall, both multiclass SVM configurations provide a good tradeoff between topology learning performance and scalability. On the contrary, the multiclass SVM using a polynomial kernel fails in providing an acceptable topology learning error in terms of P_{mis} for both types of MC simulations. Thus, the choice of the kernel function is of great significance for the multiclass SVM.

In Figures 4.15 to 4.16 and 4.17 to 4.18, the “SNR”-MC simulation and the “Network Size”-MC simulation, respectively, are shown in terms of P_{ed} and P_{mis} with respect to three differently parameterized RFCs according to Section 4.3. In particular, each RFC consists of $B = 500$ binary DTs all of maximum depth $d = 5$, $d = 7$ and $d = 10$, respectively.

In Figure 4.15, the percentage of erroneously detected edges in the sense of P_{ed} according to Eq. (3.38) is displayed against the SNR. Over the entire SNR range, the error P_{ed} is very low for all considered RFC configurations. Especially in the high SNR

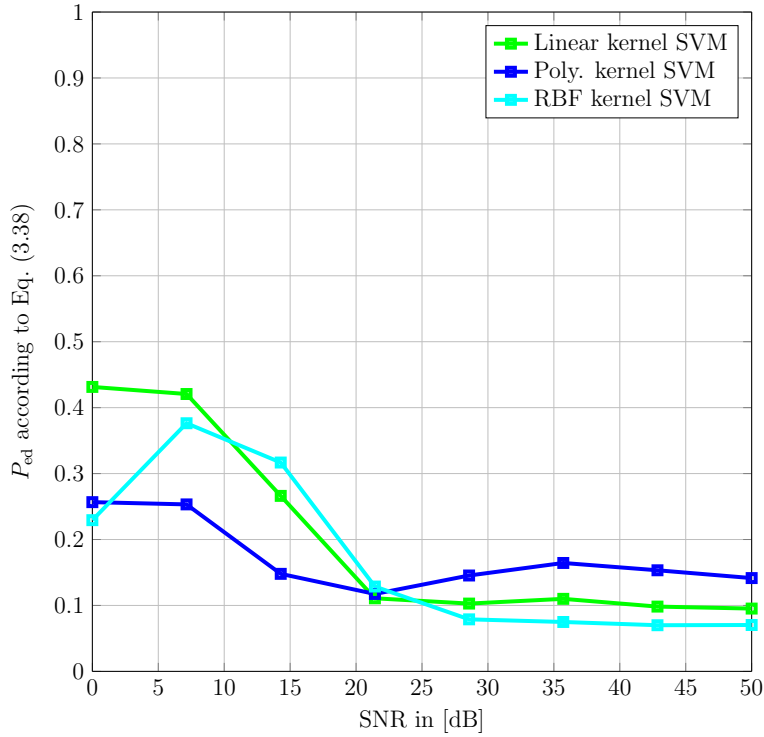


Figure 4.11. P_{ed} vs. SNR; Comparison of different SVM kernels; Polynomial: $d_r : 3$; RBF: $\gamma = 9$

regime, the error in terms of P_{ed} tends to zero. Figure 4.16 displays the error in terms of P_{mis} of the considered RFCs against the SNR. Starting from a high error P_{mis} in the low SNR regime, the considered RFCs show a constantly decreasing error rate P_{mis} with the SNR increasing. At the SNRs, the error rates P_{mis} are situated between 22% and 12%. Furthermore, the considered RFCs performance in terms of P_{ed} and P_{mis} , respectively, against the SNR is close.

In Figure 4.17, the performance of the considered RFC configurations in terms of P_{ed} against the network size is displayed at 30 dB SNR. Over the entire range of considered network sizes, the RFC configurations under study show comparable error rates P_{ed} which are remarkably low and roughly 4% only. The rate of normalized missing edges, i.e., P_{mis} , for all considered RFCs against the network size is displayed in Figure 4.18. Here, the difference in performance is obvious where the RFC with the largest depth, i.e., $d = 10$, performs best. Note that the RFCs' performance in terms of P_{mis} is much worse than their performance in terms of P_{ed} . Furthermore, P_{mis} increases with the network size increasing.

In essence, all considered RFC configurations perform well in terms of P_{ed} and P_{mis} , respectively, against a varying SNR and a changing network size, respectively. The

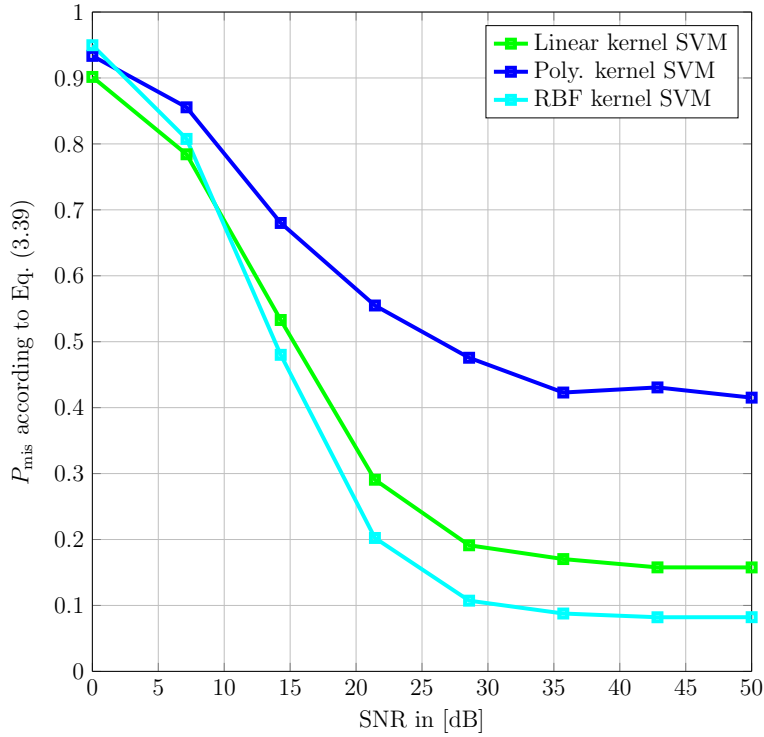


Figure 4.12. P_{mis} vs. SNR; Comparison of different SVM kernels; Polynomial: $d_r : 3$; RBF: $\gamma = 9$

difference in overall performance, with respect to the choice of the binary DT depth d , is not substantial. Although trained on smaller networks, the good estimation performance also translates to larger networks of 12 genes and more which saves a lot of computation costs during the training phase. In summary, the considered RFC configurations provide a good tradeoff between topology learning performance and scalability.

In Figures 4.19 to 4.20 and 4.21 to 4.22, the “SNR”-MC simulation and the “Network Size”-MC simulation, respectively, are shown in terms of P_{ed} and P_{mis} with respect to three ANNs according to Section 4.4 with different neural functions $f_n(\cdot)$, i.e., the logistic function, Relu function and the hyperbolic tangent function, respectively. Furthermore, each ANN is of the same network architecture that consists of 10 input nodes/neurons, a first hidden layer of 10 nodes/neurons, a second hidden layer of 10 nodes/neurons, a third hidden layer of 5 neurons and an output layer of 5 nodes/neurons.

In Figure 4.19, the percentage of erroneously detected edges according to P_{ed} described in Eq. (3.38) is displayed against the SNR. Over the entire SNR range, the error performance of the considered ANN configurations in terms of P_{ed} is very close to each

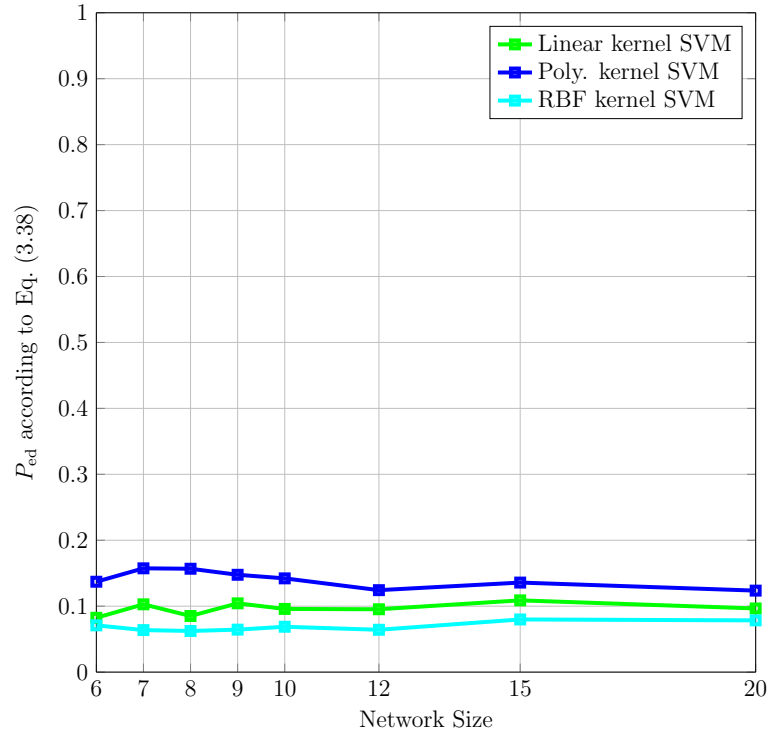


Figure 4.13. P_{ed} vs. network size; Comparison of different SVM kernels; Polynomial: $d_r : 3$; RBF: $\gamma = 9$

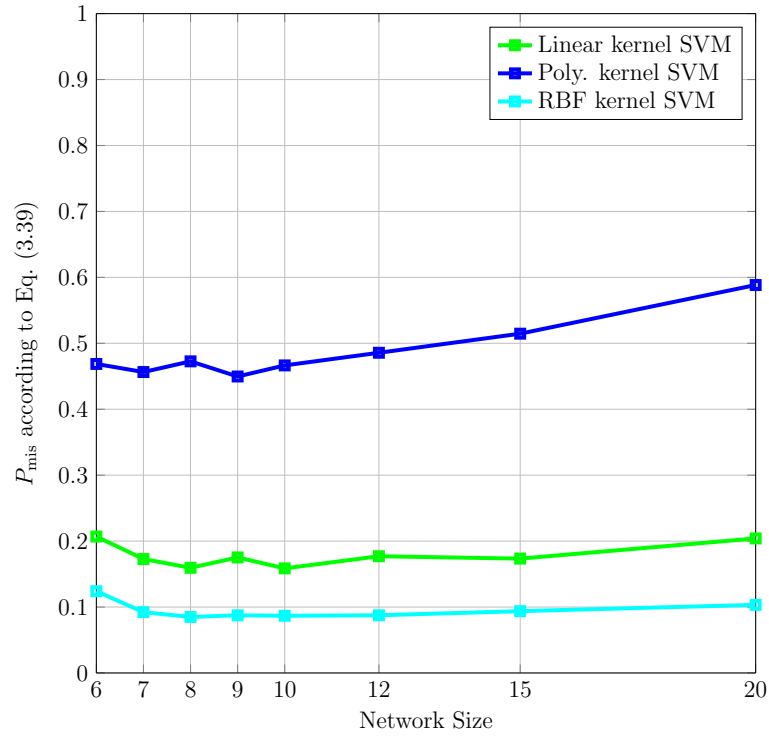


Figure 4.14. P_{mis} vs. network size; Comparison of different SVM kernels; Polynomial: $d_r : 3$; RBF: $\gamma = 9$

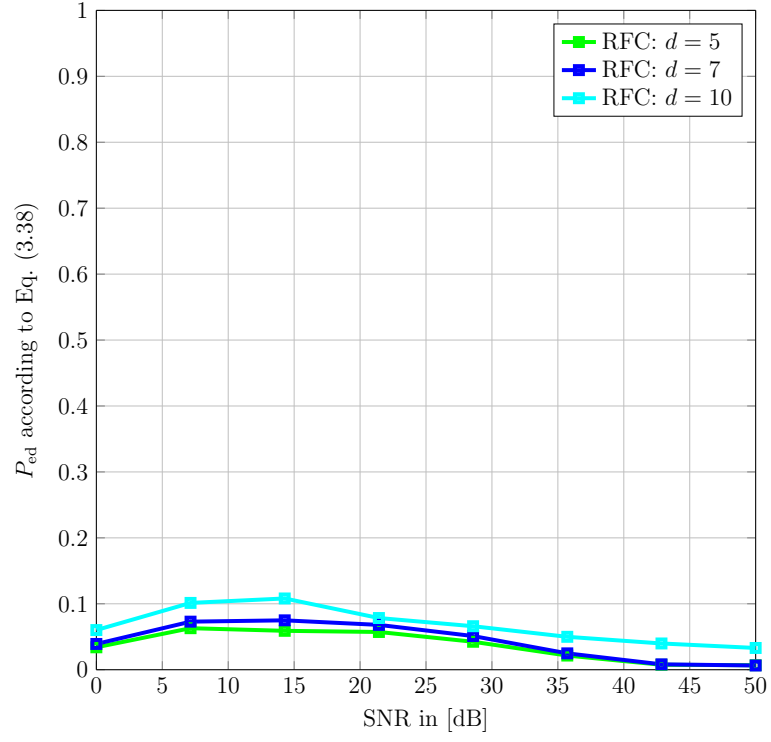


Figure 4.15. P_{ed} vs. SNR; Comparison of different RFC depths; $B = 500$

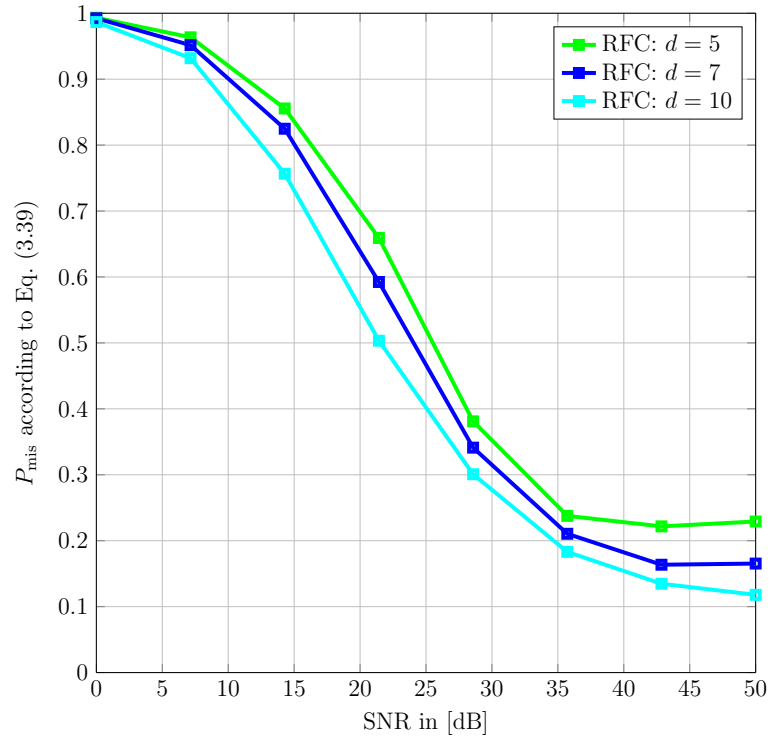


Figure 4.16. P_{mis} vs. SNR; Comparison of different RFC depths; $B = 500$

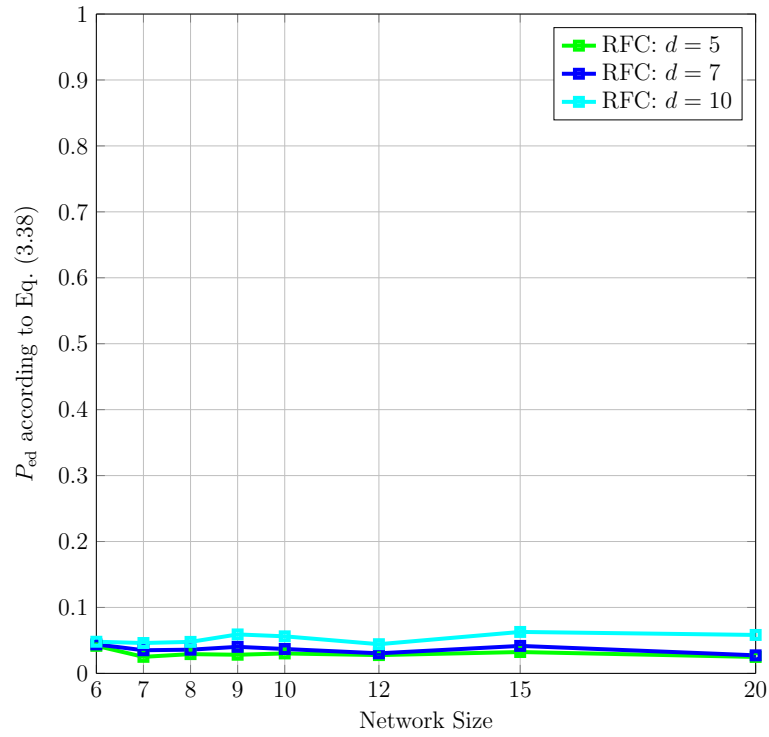


Figure 4.17. P_{ed} vs. network size; Comparison of different RFC depths; $B = 500$

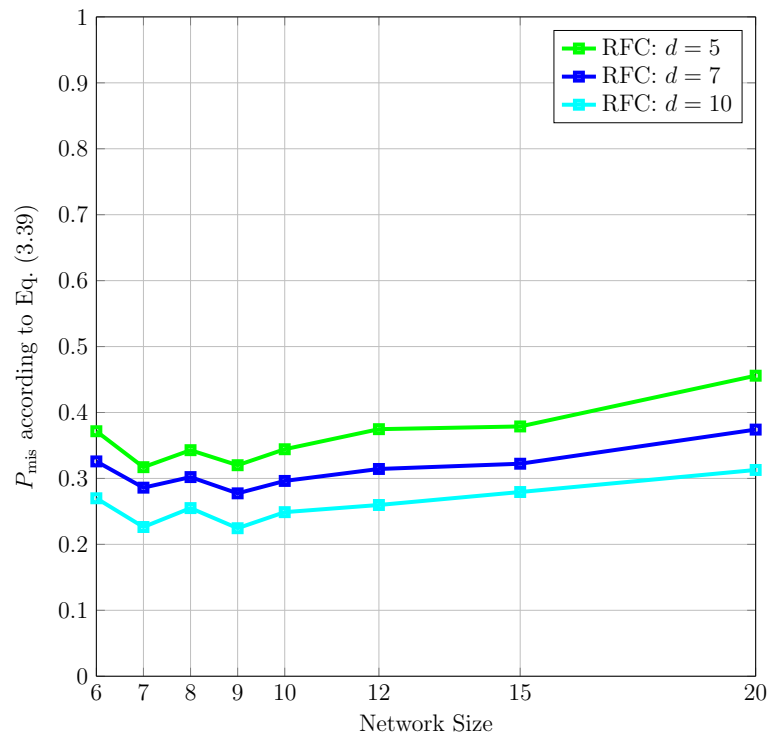


Figure 4.18. P_{mis} vs. network size; Comparison of different RFC depths; $B = 500$

other, converging to an error P_{ed} of almost zero in the high SNR regime. Figure 4.20 displays the error in terms of P_{mis} of the considered ANNs against the SNR. Starting from a high error P_{mis} in the low SNR region, the considered ANNs show a constantly decreasing error rate P_{mis} with the SNR increasing. Moreover, at SNRs above 40 dB, the error rates P_{mis} of the considered ANN configurations are situated between 3% and 5%, respectively, which depicts a remarkably good performance.

In Figure 4.21, the performance of the considered ANN configurations in terms of P_{ed} against the network size is displayed at 30 dB SNR. Over the entire range of considered network sizes, the ANN configurations under study show comparable error rates P_{ed} which are remarkably low and roughly 5% only. The rate of normalized missing edges P_{mis} displayed in Figure 4.22 shows again that the considered ANN configurations perform approximately equally well over the entire range of network sizes with a substantially low error P_{mis} of around 5%.

Essentially, all considered ANN configurations perform well in terms of P_{ed} and P_{mis} , respectively, against a varying SNR and a changing network size, respectively. Furthermore, the difference in overall performance, with respect to the choice of the neural function $f_{\text{n}}(\cdot)$, is neglectable. Identically to the multiclass SVMs and the RFCs, the considered ANNs are trained on smaller networks only, but they are still able to translate the good estimation performance to larger networks of 12 genes and more saving a lot of computational costs during the training phase. Moreover, the considered ANN configurations yield a very good tradeoff between network topology learning performance and scalability.

It is an interesting observation that all considered ML algorithms show a substantially low error in terms of erroneously estimated edges, i.e., P_{ed} , for both types of MC simulations. However, this can be explained by considering the implications of the empirical distribution of the labels y according to Table 4.4. Loosely speaking, during the training procedures the considered ML classifiers are adapted to the training set \mathcal{T} in the sense that their parameters are determined to map as many feature vectors $\mathbf{x} \in \mathcal{T}$ as possible to their correct labels y . Hence, the considered ML classifiers are only well trained if feature vectors corresponding to class $y = 0$, i.e., “no edge present”, are correctly mapped to this class label, due to the heavy dominance of class $y = 0$ in the data. In other words, the considered ML algorithms predominantly learned to reliably identify the case when no edge is present. Consequently, this leads to the described strong performance in terms of P_{ed} .

Finally, in Figures 4.23 to 4.24 and 4.25 to 4.26, the multiclass SVM with a RBF kernel, the RFC with $B = 500$ binary DTs of maximal depth $d = 10$ as well as the ANN

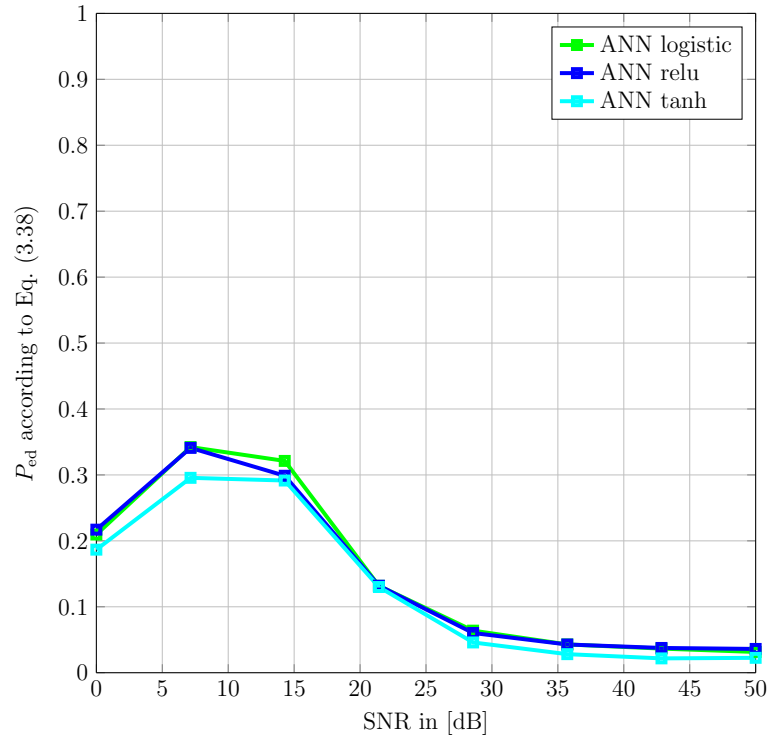


Figure 4.19. P_{ed} vs. SNR; ANN comparison using different neural functions

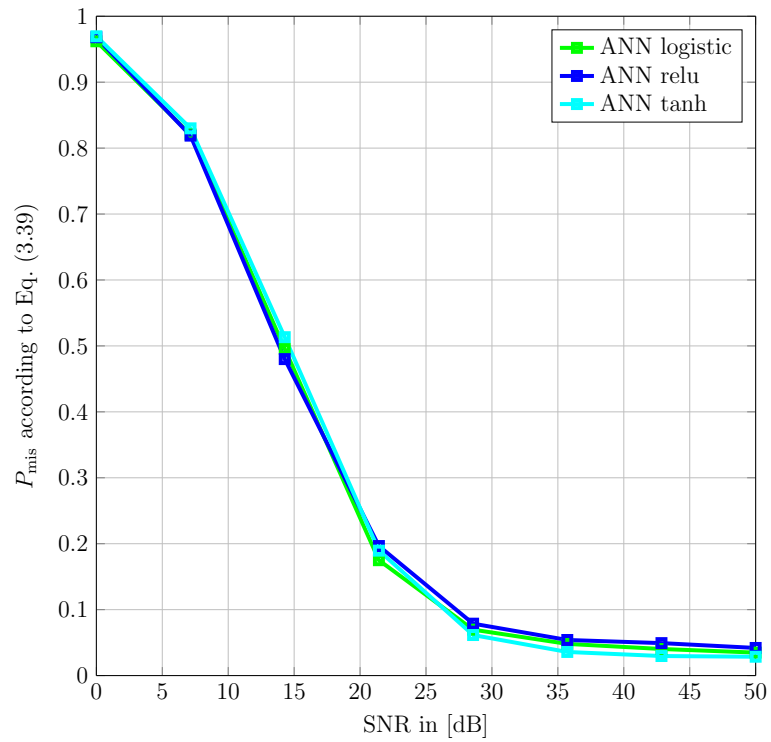


Figure 4.20. P_{mis} vs. SNR; ANN comparison using different neural functions

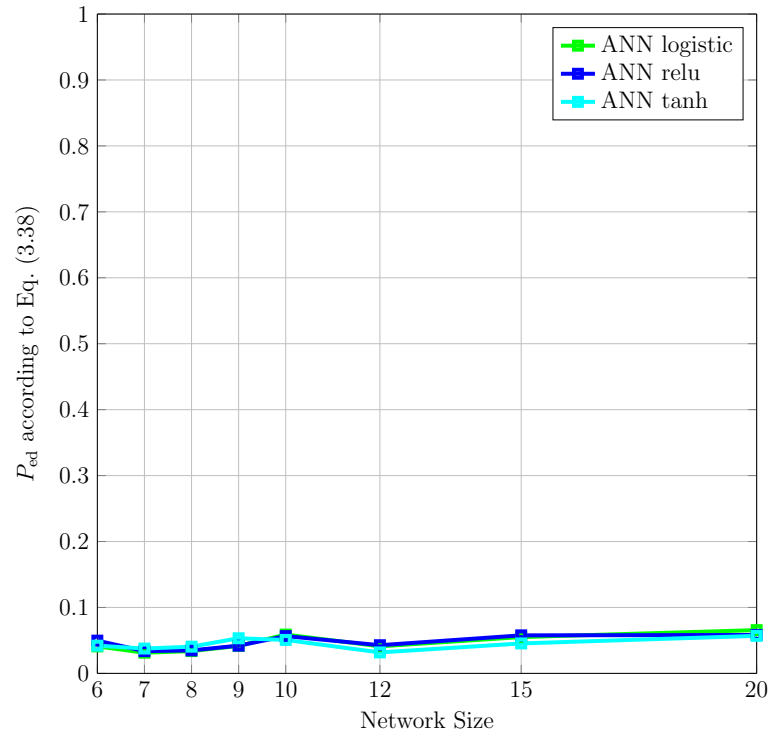


Figure 4.21. P_{ed} vs. network size; ANN comparison using different neural functions

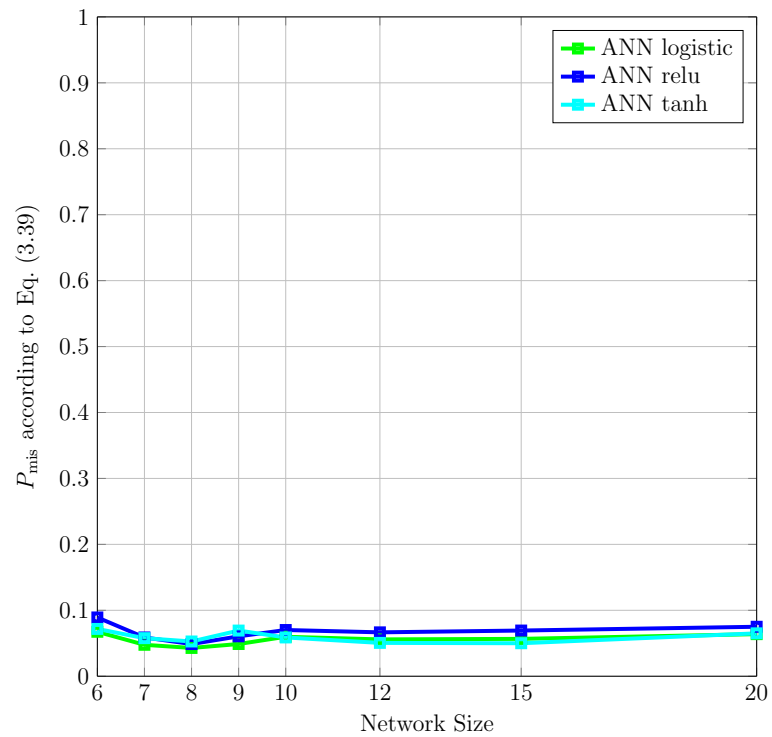


Figure 4.22. P_{mis} vs. network size; ANN comparison using different neural functions

classifier with a logistic neural function are compared to the proposed GENIE and GI-GENIE algorithms for both types of MC simulations. In particular, in Figures 4.23 to 4.24 the considered ML algorithms are compared to the GENIE/GI-GENIE algorithms in terms of P_{ed} versus the SNR. In the low SNR regime, the considered ML classifiers outperform the GENIE/GI-GENIE algorithms. However, for SNRs above 30 dB the GI-GENIE algorithm outperforms all competing methods. Interestingly, the GENIE algorithm shows an inferior performance over almost the entire SNR region. Figures 4.25 to 4.26 show the considered algorithms' performance in terms of P_{mis} against the SNR. Here, the GI-GENIE algorithm outperforms the other methods over the entire range of SNR. In this case, the GENIE algorithm shows a solid performance being superior to the considered ML classifiers in the low and high SNR region.

In Figures 4.25 to 4.26, the considered algorithms are compared in terms of P_{ed} and P_{mis} with respect to a changing network size. As depicted in Figure 4.25, all considered methods show a trend of an increasing error P_{ed} with the network size increasing. Interestingly, the error P_{ed} almost doubles in the case of the GENIE algorithm, whereas the remaining methods only exhibit a moderate error increase. Note that above a DAG size of more than 15 genes the GI-GENIE algorithm could not solve one instance of problem (3.23) within the set time limit that was one hour. Hence, for such networks the GI-GENIE's performance in terms of P_{ed} has been set to 1 in order to indicate that the considered problem instance could not be solved. Turning towards Figure 4.26, all considered methods show a trend of a slightly increasing error P_{mis} . Note again that the GI-GENIE algorithm has not been able to solve one problem instance of (3.23) for DAGs bigger than 15 genes within the set time limit, indicated by an error in terms of P_{mis} of 1.

Despite the ML algorithms' considerable performance in terms of P_{ed} and P_{mis} , respectively, against the SNR and a changing network size, they cannot guarantee that their estimated DAG topologies fulfill the structure requirements according to [BJW⁺10] as discussed in Section 2.2 which is a major shortcoming of those off-the-shelf ML classifiers.

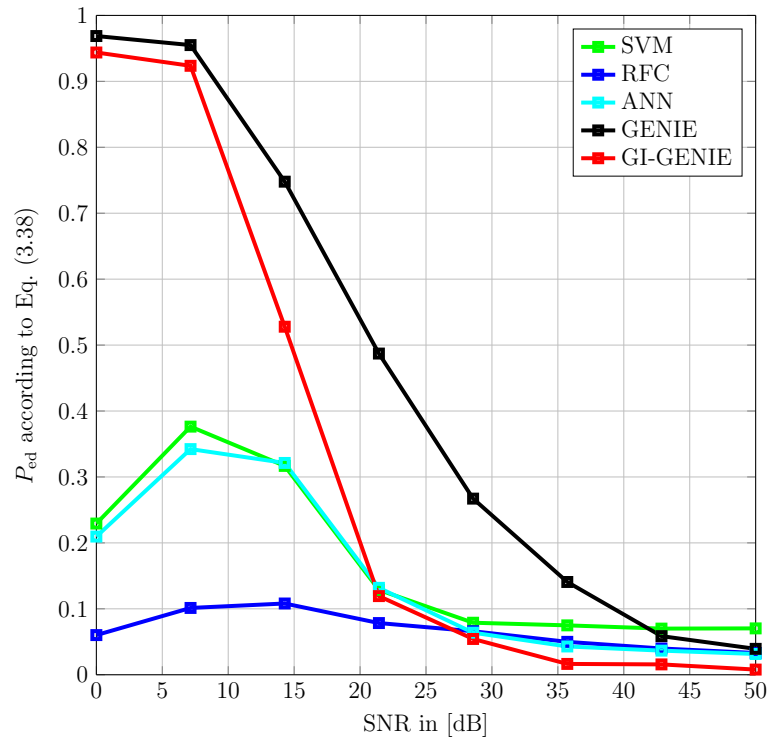


Figure 4.23. P_{ed} vs. SNR; GENIE/GI-GENIE vs. ML algorithms; SVM: RBF Kernel: $\gamma = 9$; RFC: depth $d = 10$, DT number $B = 500$; ANN: logistic neural function; $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.85$

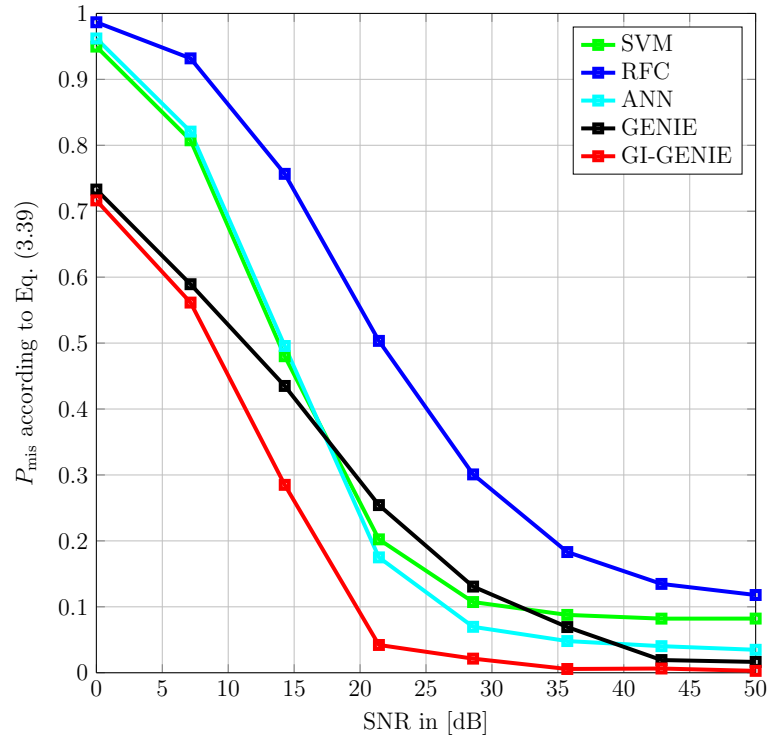


Figure 4.24. P_{mis} vs. SNR; GENIE/GI-GENIE vs. ML algorithms; SVM: RBF Kernel: $\gamma = 9$; RFC: depth $d = 10$, DT number $B = 500$; ANN: logistic neural function; $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.85$

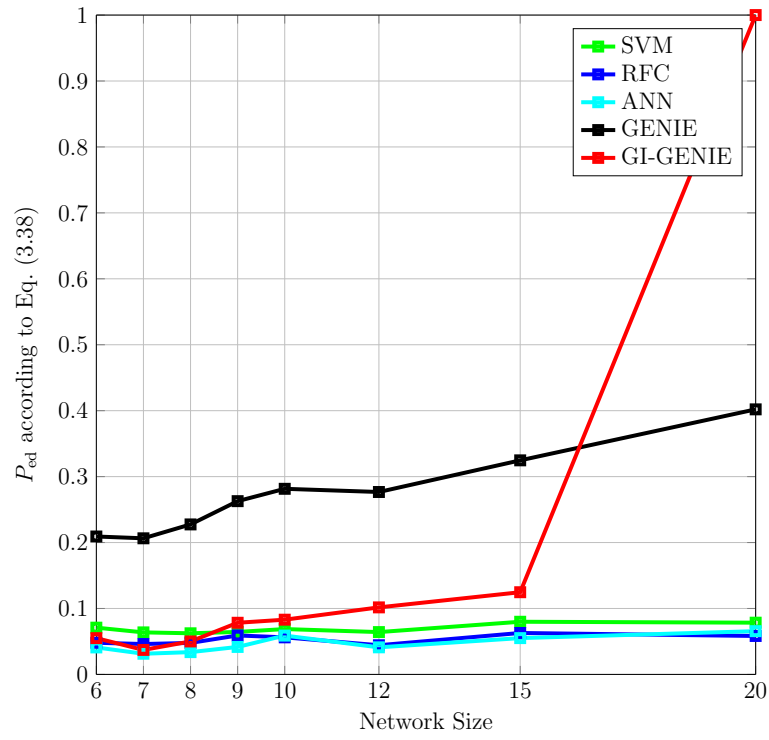


Figure 4.25. P_{ed} vs. Network Size; GENIE/GI-GENIE vs. ML algorithms; SVM: RBF Kernel: $\gamma = 9$; RFC: depth $d = 10$, DT number $B = 500$; ANN: logistic neural function; $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.85$

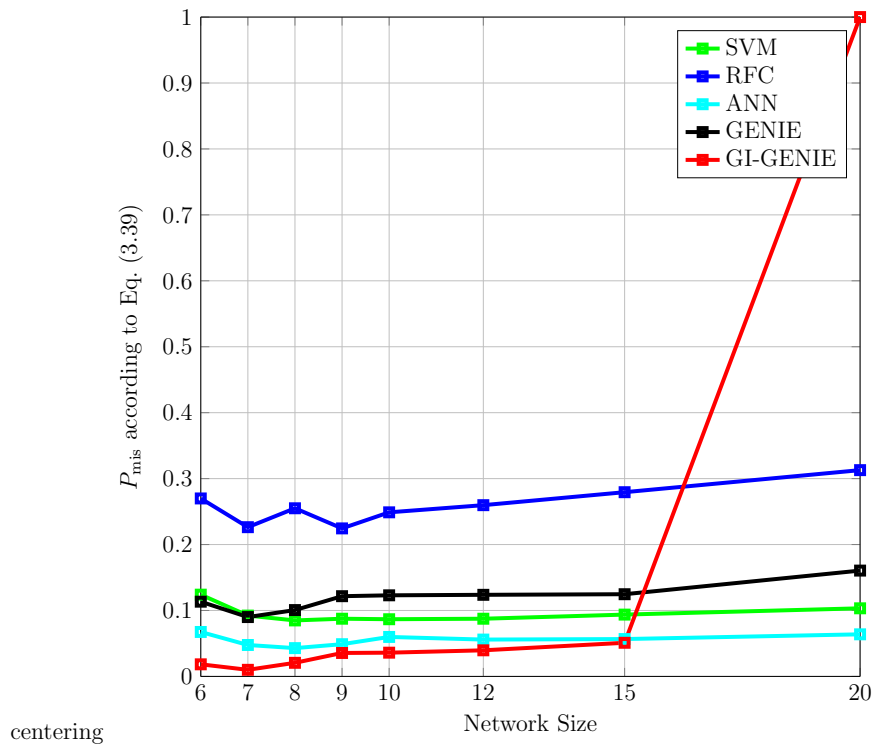


Figure 4.26. P_{mis} vs. Network Size; GENIE/GI-GENIE vs. ML algorithms; SVM: RBF Kernel: $\gamma = 9$; RFC: depth $d = 10$, DT number $B = 500$; ANN: logistic neural function; $\lambda_d = 10$, $\lambda_c = 1$, $\lambda_p = 0.85$

4.6 Summary

In this chapter, selected ML algorithms for multi-categorical classification have been presented in order to learn GI-networks in their hierarchical relationship class representation as well as in their graph representation, respectively.

In Section 4.1, two different feature engineering approaches for DAG learning are presented. The first feature vector design approach is dedicated to learning DAGs in their hierarchical relationship class representation, whereas the second feature vector design approach aims at directly learning DAG topologies in graph domain.

Furthermore, in Section 4.2 to Section 4.4 the considered ML classifiers are briefly presented that are the multiclass SVM, the RFC as well as the ANN.

In Section 4.5, the performance of the considered ML classifiers is evaluated and compared to the ILP formulation based algorithms, i.e., the GENIE and the GI-GENIE algorithm, in terms of two types of MC simulations on synthetic data. In the first type of MC simulations, the algorithms' performance is evaluated against a changing SNR with a fixed considered network size. On the contrary, in the second type of MC simulations the algorithms' performance is evaluated against a changing network size with a fixed SNR.

In particular, in Subsection 4.5.1 the considered ML classifiers are utilized to learn the DAGs in their hierarchical relationship class representation based on the feature vector design proposed in Subsection 4.1.1. The performance measure used in both cases of MC simulation types is the accuracy score. Different parameterizations of the considered ML classifiers are investigated in order to evaluate their impact on the estimation performance. Furthermore, the proposed GENIE/GI-GENIE algorithms are compared to the considered ML classifiers in terms of hierarchical class learning performance. In essence, the considered ML classifiers using the feature design according to Subsection 4.1.1 provide a good tradeoff between hierarchical relationship class learning accuracy and applicability to large-scale networks. However, it is worth mentioning that the presented ML classifiers cannot ensure that their respective sets of estimated hierarchical relationship classes encode a GI-network of the desired structure under the terms of [BJW⁺10].

In Subsection 4.5.2 the considered ML classifiers are utilized to learn the GI-network topology in graph domain based on the proposed feature design of Subsection 4.1.2.

The metrics used in both cases of MC simulation types to evaluate the topology learning performance are P_{ed} according to Eq. (3.38) and P_{mis} according to Eq. (3.39), respectively. Again, different parameter choices for the considered ML classifiers are investigated to evaluate their impact on the estimation performance. Finally, the proposed GENIE/GI-GENIE algorithms are compared to the considered ML classifiers in terms of P_{ed} and P_{mis} for both types of MC simulations. The ML classifiers' performance in terms of P_{ed} and P_{mis} for both types of MC simulations is remarkably well. Again, it is important to note that none of the considered ML classifiers can guarantee that their estimates are of the desired DAG topology. Nevertheless, the considered ML algorithms for classification yield a considerable tradeoff in terms of topology learning performance and scalability.

Chapter 5

Conclusions and Outlook

In this dissertation, several graph learning algorithms, especially customized for estimating GI-networks of a desired structure, are proposed. The methods proposed throughout this thesis can be essentially divided into the following two types: first, ILP based graph learning algorithms and secondly, general off-the-shelf ML algorithms. In particular, the proposed ILP based graph learning algorithms directly incorporate the regularities of the considered biological data model into the learning procedure. Furthermore, this data model incorporation provides certain structure guarantees with respect to the learned GI-network topology. On the contrary, the proposed graph learning methods, which are based on off-the-shelf ML techniques, do not intrinsically incorporate the system knowledge of the considered biological data model and hence, cannot provide any guarantee that the learned GI-network is always of the desired structure.

One of the major contributions of this work is the proposed GENIE algorithm that makes use of SK and DK data. As an ILP formulation based method, it is able to model the complex dependencies, which arise from the considered biological data model, in order to enhance the topology learning performance and to guarantee the desired network structure. The second proposed ILP formulation based method is the GI-GENIE algorithm. As an extension to the proposed GENIE method, the GI-GENIE algorithm addresses one significant shortcoming of the GENIE algorithm, that is the use of a single data type only, by incorporating multiple data types into the graph learning procedure. Due to their combinatorial nature, the proposed GENIE algorithm as well as the proposed GI-GENIE algorithm, respectively, suffer from a limited capability of learning large scale GI-networks. To overcome this deficit, the SEQSCA method is developed and serves as a broader framework for any type of graph learning method that is dedicated to the special structure of the considered GI-networks. It is worth mentioning that the interaction network learned by the SEQSCA algorithm is not of the originally desired GI-network structure, but of a different nature instead.

In order to assess the performance of the proposed ILP formulation based methods, they are compared to a heuristic algorithm based on a customized variant of the well known AIS method that is also capable of ensuring the desired structure of the GI-network estimates. For this purpose, MC simulations on synthetic data have been conducted. Furthermore, the proposed ILP formulation based methods, along with the

mentioned benchmark method, are applied to a large scale set of real data by means of the SEQSCA algorithm. The following observations are worth mentioning:

- In the case of the conducted MC simulations, the considered variant of the AIS method shows robust estimation results in the noisy data regime, however it is clearly outperformed by the proposed ILP formulation based algorithms over the entire range of considered SNR scenarios.
- The presented real data results of the ILP formulation based algorithms and the considered AIS method demonstrate that the GENIE/GI-GENIE methods are superior to the considered AIS variant in terms of the mentioned important biological aspects.

As the proposed ILP formulation based algorithms naturally scale badly with the network size, well known off-the-shelf ML algorithms for classification are trained with two different types of feature vector designs, respectively, in order to learn GI-networks of the desired structure. In particular, the considered ML classifiers, including the multi-class SVM, the RFC and the ANN, are trained on data according to the first feature design that aims at estimating GI-networks in the domain of the hierarchical relationship classes. In addition to that, the considered ML classifiers are also trained with respect to the second feature design which aims at estimating the GI-networks in terms of their sets of directed edges, i.e., to directly estimate the GI-network topology. For both feature designs, the investigated standard ML classifiers are compared to the GENIE/GI-GENIE methods by MC simulations against the SNR and the network size, respectively. The following observations are made:

- In the case of the first feature design approach, the considered ML classifiers exhibit a good estimation accuracy in the SNR MC simulations, though they are clearly inferior to the proposed ILP formulation based algorithms. In the network size MC simulations, the proposed ILP based algorithms show their limitations in terms of applicability to large networks, whereas the considered ML algorithms are able to manage large network sizes well. In essence, the considered ML algorithms provide a good tradeoff between estimation quality and scalability in terms of the network size.
- In the case of the second feature design approach, the same observations as for the first feature design approach can be made. For the second feature design approach, the considered ML classifiers perform fairly well in both types of MC simulations, partly outperforming the proposed GENIE algorithm. Hence, the

considered ML algorithms provide a good tradeoff between estimation quality and scalability in terms of the network size.

- However, it is worth to mention that the considered ML classifiers cannot guarantee the desired structure of the GI-networks in the estimates which is a major shortcoming of those methods.

Several interesting directions are envisioned in order to extend the ideas presented in this thesis. A selection of those is listed below:

- The impact of the weighting parameters in the GI-GENIE problem for the estimation outcome is considerable. Therefore, a promising direction to enhance the GI-GENIE's estimation performance is to devise new strategies of selecting those parameter.
- As illustrated in detail throughout this work, the proposed ILP formulation based algorithms can hardly be applied to estimate large scale GI-networks due to their combinatorial nature. Research could be carried out in order to reformulate the GENIE/GI-GENIE problems in a computationally more tractable fashion to make the GENIE/GI-GENIE algorithms applicable to large networks. A possible direction is to reformulate the ILPs as linear programs and attach a post-processing procedure to meet the original requirements.
- Although performing well in terms of estimation quality and scalability with respect to the network size, the considered ML classifiers cannot ensure that their estimates are of the desired GI-network structure. A post-processing procedure could be developed to remedy this drawback.

Appendix

A.1 Appendix A

The Lemma is proofed by contradiction.

Assume, that by applying an appropriate ordering procedure, the reconstruction $\hat{\mathcal{D}}$ of \mathcal{D} based on Ω_0 does not fulfill DAG-family-equivalence, i.e., $\hat{\mathcal{D}} \notin \mathcal{F}_{\Omega_0}$. In this case, $\Omega(\hat{\mathcal{D}}) \neq \Omega_0$. However, this states a contradiction, since $\hat{\mathcal{D}}$ has been derived from Ω_0 .

Hence, $\hat{\mathcal{D}} \in \mathcal{F}_{\Omega_0}$ must be true.

A.2 Appendix B - Full set of class coupling constraints \mathcal{L}

$$\mathcal{L}_1 = \left\{ \begin{array}{ll} \alpha_{1,j,l} + \alpha_{2,j,l} \geq \alpha_{1,i,j} + \alpha_{1,i,l} - 1 & (\text{A.1a}) \\ \alpha_{2,j,l} \geq \alpha_{1,i,j} + \alpha_{2,i,l} - 1 & (\text{A.1b}) \\ \alpha_{2,j,l} + \alpha_{3,j,l} + \alpha_{5,j,l} \geq \alpha_{1,i,j} + \alpha_{3,i,l} - 1 & (\text{A.1c}) \\ \alpha_{2,j,l} + \alpha_{4,j,l} \geq \alpha_{1,i,j} + \alpha_{4,i,l} - 1 & (\text{A.1d}) \\ \alpha_{5,j,l} + \alpha_{2,j,l} \geq \alpha_{1,i,j} + \alpha_{5,i,l} - 1 & (\text{A.1e}) \\ \} \forall i, j, l \in \mathcal{G} : l > j > i \end{array} \right.$$

$$\mathcal{L}_2 = \left\{ \begin{array}{ll} \alpha_{1,j,l} \geq \alpha_{2,i,j} + \alpha_{1,i,l} - 1 & (\text{A.2a}) \\ \alpha_{3,j,l} \geq \alpha_{2,i,j} + \alpha_{3,i,l} - 1 & (\text{A.2b}) \\ \alpha_{4,j,l} \geq \alpha_{2,i,j} + \alpha_{4,i,l} - 1 & (\text{A.2c}) \\ \alpha_{3,j,l} + \alpha_{5,j,l} \geq \alpha_{2,i,j} + \alpha_{5,i,l} - 1 & (\text{A.2d}) \\ \} \forall \{i, j, l\} \in \mathcal{G}_{\mathcal{D}} : l > j > i \end{array} \right.$$

$$\mathcal{L}_3 = \left\{ \begin{array}{l} \alpha_{1,j,l} + \alpha_{3,j,l} + \alpha_{4,j,l} \geq \alpha_{3,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} - 1 \\ \alpha_{3,j,l} \geq \alpha_{3,i,j} + \alpha_{2,i,l} - 1 \\ \alpha_{3,j,l} + \alpha_{5,j,l} \geq \alpha_{3,i,j} + \alpha_{5,i,l} - 1 \end{array} \right. \quad \begin{array}{l} \text{(A.3a)} \\ \text{(A.3b)} \\ \text{(A.3c)} \end{array}$$

$$\left. \vphantom{\mathcal{L}_3} \right\} \forall \{i, j, l\} \in \mathcal{G}_{\mathcal{D}} : l > j > i$$

$$\mathcal{L}_4 = \left\{ \begin{array}{l} \alpha_{1,j,l} + \alpha_{5,j,l} \geq \alpha_{4,i,j} + \alpha_{1,i,l} - 1 \\ \alpha_{5,j,l} \geq \alpha_{4,i,j} + \alpha_{2,i,l} - 1 \\ \alpha_{2,j,l} + \alpha_{3,j,l} + \alpha_{5,j,l} \geq \alpha_{4,i,j} + \alpha_{3,i,l} - 1 \\ \alpha_{5,j,l} \geq \alpha_{4,i,j} + \alpha_{5,i,l} - 1 \end{array} \right. \quad \begin{array}{l} \text{(A.4a)} \\ \text{(A.4b)} \\ \text{(A.4c)} \\ \text{(A.4d)} \end{array}$$

$$\left. \vphantom{\mathcal{L}_4} \right\} \forall \{i, j, l\} \in \mathcal{G}_{\mathcal{D}} : l > j > i$$

$$\mathcal{L}_5 = \left\{ \begin{array}{l} \alpha_{1,j,l} + \alpha_{4,j,l} \geq \alpha_{5,i,j} + \alpha_{1,i,l} - 1 \\ \alpha_{3,j,l} + \alpha_{4,j,l} \geq \alpha_{5,i,j} + \alpha_{2,i,l} + \alpha_{3,i,l} - 1 \\ \alpha_{4,j,l} \geq \alpha_{5,i,j} + \alpha_{4,i,l} - 1 \end{array} \right. \quad \begin{array}{l} \text{(A.5a)} \\ \text{(A.5b)} \\ \text{(A.5c)} \end{array}$$

$$\left. \vphantom{\mathcal{L}_5} \right\} \forall \{i, j, l\} \in \mathcal{G}_{\mathcal{D}} : l > j > i$$

A.3 Appendix C - Full set of DAG-topology constraints \mathcal{L}_c

$$\mathcal{L}_{c,1} = \left\{ \begin{array}{ll} 1 - \beta_{i,j} \geq \alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} - 2 & (\text{A.6a}) \\ \frac{1}{2}(\alpha_{1,i,l} + \alpha_{2,j,l}) \geq \alpha_{1,i,j} - z_{l,i,j} & (\text{A.6b}) \\ 1 - \beta_{i,j} + q_{i,j} \geq \alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} - 2 & (\text{A.6c}) \\ \frac{1}{2}(\alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l}) \geq \alpha_{4,i,j} - z_{l,i,j} - q_{i,j} & (\text{A.6d}) \\ 2 - \beta_{i,j} - q_{i,j} \geq \alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l} - 2 & (\text{A.6e}) \\ \frac{1}{2}(\alpha_{1,i,l} + \alpha_{2,j,l} + \alpha_{5,j,l}) \geq \alpha_{4,i,j} - z_{l,i,j} - 1 + q_{i,j} & (\text{A.6f}) \\ 1 - \beta_{i,j} \geq \alpha_{2,i,j} + \alpha_{2,i,l} + \alpha_{1,j,l} - 2 & \\ \frac{1}{2}(\alpha_{2,i,l} + \alpha_{1,j,l}) \geq \alpha_{2,i,j} - z_{l,i,j} & (\text{A.6g}) \\ 1 - \beta_{i,j} + q_{i,j} \geq \alpha_{5,i,j} + \alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{1,j,l} + \alpha_{4,j,l} - 2 & \\ \frac{1}{2}(\alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{1,j,l} + \alpha_{4,j,l}) \geq \alpha_{5,i,j} - z_{l,i,j} - q_{i,j} & (\text{A.6h}) \\ 2 - \beta_{i,j} - q_{i,j} \geq \alpha_{5,i,j} + \alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{1,j,l} - 2 & \\ \frac{1}{2}(\alpha_{2,i,l} + \alpha_{1,j,l} + \alpha_{5,i,l}) \geq \alpha_{5,i,j} - z_{l,i,j} - 1 + q_{i,j} & (\text{A.6i}) \\ \left. \right\} \forall \{i, j, l\} \in \mathcal{G} : l > j > i \end{array}$$

$$\mathcal{L}_{c,2} = \left\{ \begin{array}{ll} 1 - \beta_{i,j} \geq \alpha_{1,i,j} + \alpha_{2,l,i} + \alpha_{1,l,j} - 2 & (\text{A.7a}) \\ \frac{1}{2}(\alpha_{2,l,i} + \alpha_{1,l,j}) \geq \alpha_{1,i,j} - z_{l,i,j} & (\text{A.7b}) \\ 1 - \beta_{i,j} + q_{i,j} \geq \alpha_{4,i,j} + \alpha_{2,l,i} + \alpha_{5,l,i} + \alpha_{1,l,j} + \alpha_{4,l,j} - 2 & (\text{A.7c}) \\ \frac{1}{2}(\alpha_{2,l,i} + \alpha_{5,l,i} + \alpha_{1,l,j} + \alpha_{4,l,j}) \geq \alpha_{4,i,j} - z_{l,i,j} - q_{i,j} & (\text{A.7d}) \\ 2 - \beta_{i,j} - q_{i,j} \geq \alpha_{4,i,j} + \alpha_{2,l,i} + \alpha_{1,l,j} + \alpha_{4,l,j} - 2 & (\text{A.7e}) \\ \frac{1}{2}(\alpha_{2,l,i} + \alpha_{1,l,j} + \alpha_{4,l,j}) \geq \alpha_{4,i,j} - z_{l,i,j} - 1 + q_{i,j} & (\text{A.7f}) \\ 1 - \beta_{i,j} \geq \alpha_{2,i,j} + \alpha_{1,l,i} + \alpha_{2,l,j} - 2 & \\ \frac{1}{2}(\alpha_{1,l,i} + \alpha_{2,l,j}) \geq \alpha_{2,i,j} - z_{l,i,j} & (\text{A.7g}) \\ 1 - \beta_{i,j} + q_{i,j} \geq \alpha_{5,i,j} + \alpha_{1,l,i} + \alpha_{4,l,i} + \alpha_{2,l,j} + \alpha_{5,l,j} - 2 & \\ \frac{1}{2}(\alpha_{1,l,i} + \alpha_{4,l,i} + \alpha_{2,l,j} + \alpha_{5,l,j}) \geq \alpha_{5,i,j} - z_{l,i,j} - q_{i,j} & (\text{A.7h}) \\ 2 - \beta_{i,j} - q_{i,j} \geq \alpha_{5,i,j} + \alpha_{1,l,i} + \alpha_{4,l,i} + \alpha_{2,l,j} - 2 & \\ \frac{1}{2}(\alpha_{1,l,i} + \alpha_{4,l,i} + \alpha_{2,l,j}) \geq \alpha_{5,i,j} - z_{l,i,j} - 1 + q_{i,j} & (\text{A.7i}) \\ \end{array} \right\} \forall \{i, j, l\} \in \mathcal{G} : j > i > l$$

$$\mathcal{L}_{c,3} = \left\{ \begin{array}{ll} 1 - \beta_{i,j} \geq \alpha_{1,i,j} + \alpha_{1,i,l} + \alpha_{1,l,j} - 2 & (\text{A.8a}) \\ \frac{1}{2}(\alpha_{1,i,l} + \alpha_{1,l,j}) \geq \alpha_{1,i,j} - z_{l,i,j} & (\text{A.8b}) \\ 1 - \beta_{i,j} + q_{i,j} \geq \alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{1,l,j} + \alpha_{4,l,j} - 2 & (\text{A.8c}) \\ \frac{1}{2}(\alpha_{1,i,l} + \alpha_{4,i,l} + \alpha_{1,l,j} + \alpha_{4,l,j}) \geq \alpha_{4,i,j} - z_{l,i,j} - q_{i,j} & (\text{A.8d}) \\ 2 - \beta_{i,j} - q_{i,j} \geq \alpha_{4,i,j} + \alpha_{1,i,l} + \alpha_{1,l,j} + \alpha_{4,l,j} - 2 & (\text{A.8e}) \\ \frac{1}{2}(\alpha_{1,i,l} + \alpha_{1,l,j} + \alpha_{4,l,j}) \geq \alpha_{4,i,j} - z_{l,i,j} - 1 + q_{i,j} & (\text{A.8f}) \\ 1 - \beta_{i,j} \geq \alpha_{2,i,j} + \alpha_{2,i,l} + \alpha_{2,l,j} - 2 & \\ \frac{1}{2}(\alpha_{2,i,l} + \alpha_{2,l,j}) \geq \alpha_{2,i,j} - z_{l,i,j} & (\text{A.8g}) \\ 1 - \beta_{i,j} + q_{i,j} \geq \alpha_{5,i,j} + \alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{2,l,j} + \alpha_{5,l,j} - 2 & \\ \frac{1}{2}(\alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{2,l,j} + \alpha_{5,l,j}) \geq \alpha_{5,i,j} - z_{l,i,j} - q_{i,j} & (\text{A.8h}) \\ 2 - \beta_{i,j} - q_{i,j} \geq \alpha_{5,i,j} + \alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{2,l,j} - 2 & \\ \frac{1}{2}(\alpha_{2,i,l} + \alpha_{5,i,l} + \alpha_{2,l,j}) \geq \alpha_{5,i,j} - z_{l,i,j} - 1 + q_{i,j} & (\text{A.8i}) \\ \end{array} \right\} \forall \{i, j, l\} \in \mathcal{G} : j > l > i$$

List of Acronyms

ML	Machine Learning
GL	Graph Learning
DSS	Decision Support Systems
GSP	Graph Signal Processing
PPI	Protein-Protein Interaction
GI	Genetic Interaction
PDF	Probability Densitiy Function
PDFs	Probability Densitiy Functions
CI	Conditional Independence
DAG	Directed Acyclic Graph
MRF	Markov Random Field
GRF	Gaussian Random Field
TSP	Traveling Salesman Problem
MTSP	Multiple Traveling Salesmen Problem
SVM	Support Vector Machine
ANN	Artificial Neural Network
SK	Single Knockout
DK	Double Knockout
LHS	Left-Hand-Side
RHS	Right-Hand-Side
ILP	Integer-Linear-Program
NP-hard	non-deterministic polynomial time - hard
BB	Branch-and-Bound
BC	Branch-and-Cut

GENIE	Genetic-Interactions-Detector
GI-GENIE	GI-profile data extended Genetic-Interactions-Detector
SEQSCA	Sequential Scalability Technique
SMAP	Sub-Genetic-Interactions-Map
SNR	Signal-to-Noise-Ratio
MC	Monte Carlo
MCMC	Markov-Chain-Monte-Carlo
LMM	Least-Mismatch-Model
RF	Random Forest
DT	Decision Tree
GSO	Graph Shift Operator
LS	Least-Squares
SDP	Semidefinites Program
ADMM	Alternating Direction Method of Multipliers
SEM	Structural Equation Model
VARM	Vector Autoregressive Model
PC	Partial Correlation
TV	Time-Varying

List of Symbols

Symbols

\mathcal{A}	graph of general topology
a_i	node i of graph with general topology
a_j	node j of graph with general topology
$\mathcal{E}_{\mathcal{A}}$	set of edges of graph with general topology
$\mathcal{V}_{\mathcal{A}}$	set of vertices of graph with general topology
P	a general path in a graph
\mathcal{GI}	GI-network for general biological system model
$\mathcal{E}_{\mathcal{GI}}$	set of edges/interactions among the genes for general biological system model
$\mathcal{V}_{\mathcal{GI}}$	set of genes of a GI-network for a general biological system model
\mathcal{D}	GI-network/DAG according to the biological system model of [BJW ⁺ 10]
\mathbb{D}	space of all possible DAGs under the terms of [BJW ⁺ 10]
$\mathcal{E}_{\mathcal{D}}$	set of interactions of a GI-network/DAG according to [BJW ⁺ 10]
$\mathcal{V}_{\mathcal{D}}$	set of nodes/vertices of a GI-network/DAG according to [BJW ⁺ 10]
\mathcal{G}	set of genes under study
G	number of genes under study
R	reporter level of a DAG according to [BJW ⁺ 10]
$R(i)$	SK of gene i
$R(i, j)$	DK of genes i, j
$\rho(i, j)$	GI-profile measurement for gene pair i, j
\mathcal{K}	set of hierarchical relationship classes of [BJW ⁺ 10]
$\mu_k(i, j)$	expected DK phenotype of gene pair i, j being in class k
P_{τ}^i	path τ from gene i to the reporter node in a DAG according to [BJW ⁺ 10]
P_{κ}^j	path κ from gene j to the reporter node in a DAG according to [BJW ⁺ 10]
P_{ζ}^l	path ζ from gene l to the reporter node in a DAG according to [BJW ⁺ 10]
\mathcal{P}_i	set of all paths from gene i to the reporter node in a DAG according to [BJW ⁺ 10]

\mathcal{P}_j	set of all paths from gene j to the reporter node in a DAG according to [BJW ⁺ 10]
\mathcal{P}_l	set of all paths from gene l to the reporter node in a DAG according to [BJW ⁺ 10]
$\Omega(\mathcal{D})$	representation of DAG \mathcal{D} in the domain of hierarchical relationship classes of [BJW ⁺ 10]
$\omega(i, j)$	class indicator coefficient of gene pair i, j
\mathcal{F}_{Ω_0}	family/collection of all DAGs that are represented by Ω_0 in the domain of hierarchical relationship classes
$s_k(i, j)$	score to measure the mismatch to the model of [BJW ⁺ 10] for the case that gene pair i, j is classified to class k
$\alpha_{k,i,j}$	class selection variable for class k of gene pair i, j
$A_{\mathcal{D}}$	representation of DAG \mathcal{D} in the domain of class selection variables
\mathcal{L}	set of logical class coupling constraints
\mathcal{L}_k	subset of logical class coupling constraints assuming that $\alpha_{k,i,j} = 1$
$\alpha_{G,k,i,j}$	GENIE estimate of the true class selection variables $\alpha_{k,i,j}$
A_G	GENIE estimate of the true collection of class selection variables $A_{\mathcal{D}}$ of DAG \mathcal{D}
Ω_G	GENIE estimate of the true class indicator coefficient representation $\Omega(\mathcal{D})$ of DAG \mathcal{D}
\mathcal{D}_G	GENIE estimate of the true DAG topology \mathcal{D}
S_{\min}	overall mismatch score associated with the solution of the GENIE-problem
$\mathcal{I}^{(i)}$	interaction set of gene i
\mathcal{S}	SMAP in DAG \mathcal{D}
$\mathcal{N}_{\mathcal{S},\text{in}}$	set of “inward” genes of SMAP \mathcal{S}
$\mathcal{N}_{\mathcal{S},\text{out}}$	set of “outward” genes of SMAP \mathcal{S}
$\mathcal{D}_{r,k}$	relevance DAG under the assumption that $\alpha_{k,i,l} = 1$
\tilde{A}_G	redundantly expanded GENIE estimate of the true collection of class selection variables $A_{\mathcal{D}}$ of DAG \mathcal{D}
\mathcal{M}_i	set containing all genes l for which $\alpha_{k,i,l} = 1$
\mathcal{M}'_i	set containing all gene pairs $\{l, \tilde{l}\} \in \mathcal{M}_i$ that are independent of each other
P_{ed}	number of erroneously learned edges normalized by the number of “true” edges
P_{mis}	number of missing edges normalized by the number of “true” edges

P_{lin}	a linear pathway in a DAG
$P_{\text{lin,max}}$	a longest linear pathway in a DAG
\mathcal{P}_{lin}	set of all linear pathways in a DAG
$\mathcal{L}_k^{(i)}$	set of all genes l that are in class k with gene i in DAG \mathcal{D}
$L_k^{(i,j)}$	normalized number of genes in DAG \mathcal{D} that are in interaction class k with both genes i and j
$\mathbf{x}(i, j)$	feature vector of gene pair $\{i, j\}$
$\mathbf{x}_{\text{sc}}(i, j)$	part of the overall feature vector that captures the data model of [BJW ⁺ 10] of gene pair $\{i, j\}$
$\mathbf{x}_{\text{coup}}(i, j)$	part of the overall feature vector that captures the class coupling/statistical similarities of gene pair $\{i, j\}$
$y(i, j)$	corresponding feature vector label
\mathcal{T}	set of training data
$\{\mathbf{x}_t, y_t\}$	t -th training data sample
T_{r}	number of training data samples in training set \mathcal{T}
N_c	number of feature vector labels
$\mathbf{w}^{k,l}$	parameterization vector of the hyperplane separating class k from class l according to SVM design
$b^{k,l}$	shift of the hyperplane separating class k from class l according to SVM design
$\zeta^{k,l}$	auxiliary variables that allow for misclassification during the SVM design
T	binary decision tree
ν_v	split v in a binary decision tree
v	node v in binary decision tree
m	feature attribute to be split on at node
ψ	threshold value vor split
ν^L	resulting feature space of the left partition
ν^R	resulting feature space of the right partition
T^L	decision subtree corresponding to left partition
T^R	decision subtree corresponding to right partition
$(\mathbf{x})_m$	m -th attribute of feature vector \mathbf{x}
F	a random forest of binary decision trees
B	number of decision trees in a random forest
\mathcal{M}_f	a random subset of feature attributes
$\boldsymbol{\theta}$	ANN weighting coefficients
$J(\boldsymbol{\theta})$	ANN objective function

$(h_{\theta}(\cdot))_k$	value of output neuron that corresponds to class k
\tilde{y}	expanded class label
\mathcal{S}_b	subset of genes of the entire set of genes under study
$\mathcal{R}_{\mathcal{S}_b, n}$	collection of all n -dimensional knockout observations
\mathcal{R}	collection of all possible knockout observations given the gene set of interest, i.e., \mathcal{G}
n_{Ψ}	number of all possible data types that can be derived from knockout observations
\mathcal{C}	collection of all possible data types based on knockout observations
\mathfrak{M}	biological data model
$\mathbf{t}_{\mathfrak{M}}$	$n_{\mathfrak{M}}$ -dimensional tuple of knockout measurements and derived data types associated with model \mathfrak{M}
$T_{\mathfrak{M}}$	GI-network topology information associated with model \mathfrak{M}
\mathbf{z}	state vector in an simulated annealing context
T	temperature in an simulated annealing context
\mathcal{H}	set accumulating all constraints of an optimization problem
C_N	normalizing constant of a target PDF/distribution in an simulated annealing context
$\mathcal{H}_{\text{supp}}$	support of a target PDF $p(\mathbf{z})$ in an simulated annealing context
$w(\mathbf{z})$	importance weight
N_{MC}	number of Markov-chain iterations in an simulated annealing context
N_{IS}	number of samples in an importance sampling context
N_{AR}	number of annealing runs in an annealed importance sampling context

Notation

$V(\cdot)$	operator that yields the set of nodes of a graph and a path in a graph, respectively
$E(\cdot)$	operator that yields the set of edges of a graph
$ \cdot $	operator that yields the number of elements of a set
$\Omega(\cdot)$	transformation of a GI-network/DAG from graph domain to the domain of hierarchical relationship classes
$f(\cdot)$	function that maps class k to class k' according to the symmetry properties of the classes of [BJW ⁺ 10]
$\mathbf{0}^{M \times N}$	all zeros vector of dimensions $M \times N$
$\mathbf{1}^{M \times N}$	all ones vector of dimensions $M \times N$

$\phi(\cdot)$	function that maps the original feature space into a high dimensional space
$h_{\boldsymbol{\theta}}(\cdot)$	general classifier function parameterized by coefficients $\boldsymbol{\theta}$
$f_n(\cdot)$	neural activation function
$r(\cdot)$	general regulator term in ANN objective function
$\Psi_1(\cdot) - \Psi_{n_\Psi}(\cdot)$	possible knockout data transformation functions
$E(\mathbf{z})$	arbitrary objective function denoted as <i>free energy</i> in an simulated annealing context
$p(\mathbf{z})$	target PDF/distribution in an simulated annealing context
$\tilde{p}(\mathbf{z}^{(t)} \mathbf{z}^{(t-1)}, \dots, \mathbf{z}^{(0)})$	conditional PDF/distribution of state $\mathbf{z}^{(t)}$ having observed states $\mathbf{z}^{(t-1)}, \dots, \mathbf{z}^{(0)}$
$Q(\mathbf{z}^{(t)} \mathbf{z}^{(t-1)})$	transition kernel that maps from state $\mathbf{z}^{(t-1)}$ to state $\mathbf{z}^{(t)}$
$p_t(\mathbf{z}^{(t)})$	marginal distribution of $\mathbf{z}^{(t)}$ at Markov-chain iteration t
$A_c(\mathbf{z}^{(t)}, \mathbf{z}')$	probability of accepting proposal state \mathbf{z}' given state $\mathbf{z}^{(t)}$
$r(\mathbf{z}^{(t)})$	probability of staying in state $\mathbf{z}^{(t)}$
$\check{p}(\mathbf{z})$	target distribution in an importance sampling context
$\check{q}(\mathbf{z})$	proposal distribution in an importance sampling context
$f_{\text{is}}(\mathbf{z})$	function of interest in an importance sampling context
$\mathbb{E}_{\check{p}(\mathbf{z})}[\cdot]$	expected value of function $f_{\text{is}}(\mathbf{z})$ with respect to distribution $\check{p}(\mathbf{z})$

Bibliography

- [AdFDJ03] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan, “An introduction to mcmc for machine learning,” *Machine Learning*, vol. 50, no. 1, pp. 5–43, Jan 2003.
- [AG11] D. Angelosante and G. Giannakis, “Sparse graphical modeling of piecewise-stationary time series,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 1960–1963.
- [AKM05] T. Achterberg, T. Koch, and A. Martin, “Branching rules revisited,” *Operations Research Letters*, vol. 33, pp. 42–54, 2005.
- [Alp10] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, Massachusetts, London, England: The MIT Press, 2010.
- [BBG13] J. Bazerque, B. Baingana, and G. Giannakis, “Identifiability of sparse structural equation models for directed and cyclic networks,” in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 839–842.
- [Ber07] E. Berner, *Clinical decision support systems*. Springer, 2007, vol. 233.
- [BG06] M. J. Beal and Z. Ghahramani, “Variational bayesian learning of directed graphical models with hidden variables,” *Bayesian Analysis*, vol. 1, pp. 793–831, December 2006.
- [BG17] B. Baingana and G. Giannakis, “Tracking switched dynamic network topologies from information cascades,” *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 985–997, 2017.
- [BGd08] O. Banerjee, L. E. Ghaoui, and A. d’Aspremont, “Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data,” *J. Mach. Learn. Res.*, vol. 9, pp. 485–516, June 2008.
- [Bis95] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, USA: Oxford University Press, 1995.
- [BJW⁺10] A. Battle, M. C. Jonikas, P. Walter, J. S. Weissman, and D. Koller, “Automated identification of pathways from quantitative genetic interaction data,” *Molecular Systems Biology*, vol. 6, 2010.
- [BO04] A. L. Barabasi and Z. N. Oltvai, “Network biology: Understanding the cell’s functional organization,” *Nature Reviews: Genetics*, vol. 5, pp. 101–113, February 2004.
- [BPC⁺11] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

- [Bre01] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, October 2001.
- [Bri95] R. S. G. Britain), *Proceedings of the Royal Society of London*. Taylor & Francis, 1895, no. Bd. 58. [Online]. Available: <https://books.google.de/books?id=60aL0zlT-90C>
- [BV04] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [BV11] M. R. Bussieck and S. Vigerske, *MINLP Solver Software*. American Cancer Society, 2011.
- [C⁺] M. Costanzo *et al.*, “Drygin - data repository of yeast genetic interactions,” Website, <http://drygin.ccbr.utoronto.ca/~costanzo2009/>.
- [C⁺10] M. Constanzo *et al.*, “The genetic landscape of a cell,” *Science*, vol. 327, no. 5964, p. 425431, 2010.
- [CBD11] D. S. Chen, R. G. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2011.
- [CBG13] X. Cai, J. Bazerque, and G. Giannakis, “Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations,” *PLoS computational biology*, vol. 9, no. 5, p. e1003068, 2013.
- [CDPP12a] Y. Cheng, S. Drewes, A. Philipp, and M. Pesavento, “Joint network optimization and beamforming for coordinated multi-point transmission using mixed integer programming,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012, pp. 3217–3220.
- [CDPP12b] —, “Joint network topology optimization and multicell beamforming using mixed integer programming,” in *2012 International ITG Workshop on Smart Antennas (WSA)*, March 2012, pp. 187–192.
- [CGS⁺11] G. Chen, D. Glen, Z. Saad, J. Hamilton, M. Thomason, I. Gotlib, and R. Cox, “Vector autoregression, structural equation modeling, and their synthesis in neuroimaging data analysis,” *Computers in biology and medicine*, vol. 41, no. 12, pp. 1142–1155, 2011.
- [CH92] F. G. Cooper and E. Herskovits, “A bayesian method for the induction of probabilistic networks from data,” *Machine Learning*, vol. 9, pp. 309–347, October 1992.
- [Che16] J. X. Chen, “The evolution of computing: Alphago,” *Computing in Science & Engineering*, vol. 18, pp. 4–7, July-August 2016.
- [Chi96] D. M. Chickering, *Learning Bayesian Networks is NP-Complete*. New York, NY: Springer New York, 1996, pp. 121–130. [Online]. Available: https://doi.org/10.1007/978-1-4612-2404-4_12

- [CL68] C. K. Chow and C. N. Liu, “Approximating discrete probability distributions with dependence trees,” *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, May 1968.
- [CL01] A. Curtis and A. Lomax, “Prior information, sampling distributions, and the curse of dimensionality,” *GEOPHYSICS*, vol. 66, no. 2, pp. 372–378, 2001.
- [Coo16] D. Cooper, “Power module integration,” *IEEE Power Electronics Magazine*, vol. 3, pp. 31–36, September 2016.
- [CP13] Y. Cheng and M. Pesavento, “An optimal iterative algorithm for codebook-based downlink beamforming,” *IEEE Signal Processing Letters*, vol. 20, no. 8, pp. 775–778, Aug 2013.
- [CP15] ———, “Joint discrete rate adaptation and downlink beamforming using mixed integer conic programming,” *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1750–1764, April 2015.
- [CPP12] Y. Cheng, A. Philipp, and M. Pesavento, “Dynamic rate adaptation and multiuser downlink beamforming using mixed integer conic programming,” *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pp. 824–828, 2012.
- [CPP13] Y. Cheng, M. Pesavento, and A. Philipp, “Joint network optimization and downlink beamforming for comp transmissions using mixed integer conic programming,” *IEEE Transactions on Signal Processing*, vol. 61, no. 16, pp. 3972–3987, Aug 2013.
- [Cra89] S. L. Crawford, “Extensions to the cart algorithm,” *International Journal of Man-Machine Studies*, vol. 31, no. 2, pp. 197–217, 1989.
- [CS96] P. Cheeseman and J. Stutz, *Bayesian Classification (Autoclass): Theory and Results*. Cambridge, Massachusetts, London, England: The MIT Press, 1996, pp. 153–180.
- [CSMH91] J. Chang-Sung and K. Myung-Ho, “Fast parallel simulated annealing for traveling salesman problem on simd machines with linear interconnections,” *Parallel Computing*, vol. 17, no. 2, pp. 221 – 228, 1991.
- [CST00] N. Christianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*. The Pitt Building, Trumpington Street, Cambridge UK: Cambridge University Press, 2000.
- [CT76] D. Cheriton and R. E. Tarjan, “Finding minimum spanning trees,” *Society for Industrial and Applied Mathematics (SIAM) Journal on Computing*, vol. 5, pp. 724–742, December 1976.
- [CV95] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, pp. 273–297, September 1995.

- [CVSK15] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, “Discrete signal processing on graphs: Sampling theory,” *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510–6523, 2015.
- [D⁺05] B. L. Drees *et al.*, “Derivation of genetic interaction networks from quantitative phenotype data,” *Genome Biology*, vol. 6, no. 4, p. R38, Mar 2005.
- [DCB⁺09] S. J. Dixon, M. Costanzo, A. Baryshnikova, B. Andrews, and C. Boone, “Systematic mapping of genetic interaction networks,” *Annual Review of Genetics*, vol. 43, no. 1, pp. 601–625, 2009.
- [DFJ54] G. Dantzig, R. Fulkerson, and S. Johnson, “Solution of a large-scale traveling-salesman problem,” *Journal of the Operations Research Society of America*, vol. 2, pp. 393–410, 1954.
- [DGK08] J. Duchi, S. Gould, and D. Koller, “Projected subgradient methods for learning sparse gaussians,” in *Proceedings of the Twenty-fourth Conference on Uncertainty in AI (UAI)*, 2008.
- [Die06] R. Diestel, *Graphentheorie*. Berlin, Heidelberg, Deutschland: Springer Verlag, 2006.
- [Dob07] A. Dobra, “Dependency networks for genome-wide data,” *Biostatistics*, vol. 8, pp. 1–28, May 2007.
- [Don] J. R. Donoghue, “Comparison of integer programming (ip) solvers for automated test assembly (ata),” *ETS Research Report Series*, vol. 2015, no. 1, pp. 1–12. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ets2.12051>
- [DS98] N. R. Draper and H. Smith, *Applied Regression Analysis, 3rd Edition*. Hoboken, New Jersey: John Wiley & Sons, Inc., 1998.
- [DWW14] P. Danaher, P. Wang, and D. Witten, “The joint graphical lasso for inverse covariance estimation across multiple classes,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 76, no. 2, pp. 373–397, 2014.
- [FHT08] J. Friedman, T. Hastie, and R. Tibshirani, “Sparse inverse covariance estimation with the graphical lasso,” *Biostatistics (Oxford, England)*, vol. 9, pp. 432–441, July 2008.
- [Flo95] C. A. Floudas, *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.
- [Fre09] D. A. Freedman, *Statistical Models: Theory and Practice*. 32 Avenue of the Americas, New York, USA: Cambridge University Press, 2009.

- [Fri97] N. Friedman, “Learning belief networks in the presence of missing values and hidden variables,” in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML ’97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 125–133. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645526.657145>
- [FSJW09] E. Fox, E. Sudderth, M. Jordan, and A. Willsky, “Nonparametric bayesian learning of switching linear dynamical systems,” in *Advances in Neural Information Processing Systems*, 2009, pp. 457–464.
- [GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Camebridge, MA, USA: The MIT Press, 2016.
- [Gew89] J. Geweke, “Bayesian inference in econometric models using monte carlo integration,” *Econometrica*, vol. 57, no. 6, pp. 1317–1339, 1989.
- [GI89] W. Glynn and D. L. Iglehart, “Importance sampling for stochastic simulations,” *Management Science*, vol. 35, no. 11, pp. 1367–1392, 1989.
- [GLMZ11] J. Guo, E. Levina, G. Michailidis, and J. Zhu, “Joint estimation of multiple graphical models,” *Biometrika*, vol. 98, no. 1, pp. 1–15, 2011.
- [Gra69] C. Granger, “Investigating causal relations by econometric models and cross-spectral methods,” *Econometrica: Journal of the Econometric Society*, pp. 424–438, 1969.
- [GSK18] G. Giannakis, Y. Shen, and G. Karanikolas, “Topology identification and learning over graphs: Accounting for nonlinearities and dynamics,” *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.
- [Ham95] J. Hamilton, “Time series analysis: Chapter 11,” *Economic Theory. II, Princeton University Press, USA*, pp. 625–630, 1995.
- [Han10] D. J. Hand, “Statistical analysis of network data: Methods and models by eric d. kolaczyk,” *International Statistical Review*, vol. 78, no. 1, pp. 135–135, 2010.
- [Has70] W. K. Hastings, “Monte carlo sampling methods using markov chains and their applications,” *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.
- [HCM⁺00] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, “Dependency networks for inference, collaborative filtering, and data visualization,” *The Journal of Machine Learning Research*, vol. 1, pp. 49–75, September 2000.
- [HL02] C. W. Hsu and C. Lin, “A comparison of methods for multiclass support vector machines,” *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.
- [Hoc98] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, 1998.

- [HRVW96] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz, “An interior-point method for semidefinite programming,” *SIAM Journal on Optimization*, vol. 6, no. 2, pp. 342–361, 1996.
- [HS10] J. Honorio and D. Samaras, “Multi-task learning of gaussian graphical models.” in *ICML*. Citeseer, 2010, pp. 447–454.
- [J+96] F. Jensen *et al.*, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 210.
- [Joa02] T. Joachims, *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.
- [Kap08] D. Kaplan, *Structural equation modeling: Foundations and extensions*. Sage Publications, 2008, vol. 10.
- [KB97] S. Kotz and N. Balakrishnan, *Advances in Urn Models during the Past Two Decades*. Boston, MA: Birkhäuser Boston, 1997, pp. 203–257.
- [Key00] R. W. Keyes, “Miniaturization of electronics and its limits,” *Reprinted From IBM Journal of Research and Development*, vol. 44, pp. 84–88, Januar-March 2000.
- [KGSL16] G. Karanikolas, G. Giannakis, K. Slavakis, and R. Leahy, “Multi-kernel based nonlinear models for connectivity identification of brain networks,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6315–6319.
- [KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [KKL+07] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, “An interior-point method for large-scale ℓ_1 -regularized least squares,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, Dec 2007.
- [Kri15] P. O. Kristensson, “Next-generation text entry,” *IEEE Computer*, vol. 48, pp. 84–87, July 2015.
- [Kru56] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, February 1956.
- [LB+95] Y. LeCun, Y. Bengio *et al.*, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
- [LBH15] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, May 2015.

- [LBL16] Y. Lin, Z. Bian, and X. Liu, “Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing-tabu search algorithm to solve the symmetrical traveling salesman problem,” *Applied Soft Computing*, vol. 49, pp. 937 – 952, 2016.
- [LJB⁺95] Y. Lecun, L. Jackel, L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, *Learning algorithms for classification: A comparison on handwritten digit recognition*. World Scientific, 1995, pp. 261–276.
- [LW66] E. L. Lawler and D. E. Wood, “Branch-and-bound methods: A survey,” *Operations Research*, vol. 14, pp. 699–719, 1966.
- [LZS16] M. Liu, D. Zhang, and D. Shen, “Relationship induced multi-template learning for diagnosis of alzheimer’s disease and mild cognitive impairment,” *IEEE Transactions on Medical Imaging*, vol. 35, pp. 1463–1474, June 2016.
- [Ma10] W. Ma, “Semidefinite relaxation of quadratic optimization problems and applications,” *IEEE Signal Processing Magazine*, vol. 1053, no. 5888/10, 2010.
- [MB06] N. Meinshausen and P. Bühlmann, “High-dimensional graphs and variable selection with the lasso,” *The Annals of Statistics*, vol. 36, pp. 1436–1462, August 2006.
- [Meh92] S. Mehrotra, “On the implementation of a primal-dual interior point method,” *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 575–601, 1992.
- [MMWW02] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey, “Cutting planes in integer and mixed integer programming,” *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 397 – 446, 2002.
- [MNB⁺06] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano, “Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context,” *BMC Bioinformatics*, vol. 7, pp. 1–15, March 2006.
- [MRR⁺53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, “Simulated annealing,” *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.
- [MSLR16] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, “Sampling of graph signals with successive local aggregations,” *IEEE Transactions on Signal Processing*, vol. 64, no. 7, pp. 1832–1843, April 2016.
- [Mur12] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, Massachusetts, London, England: The MIT Press, 2012.

- [MVB18] D. Meira, J. Viterbo, and F. Bernardini, “An experimental analysis on scalable implementations of the alternating least squares algorithm,” in *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2018, pp. 351–359.
- [Nea01] R. M. Neal, “Annealed importance sampling,” *Statistics and Computing*, vol. 11, no. 2, pp. 125–139, 2001.
- [NP16] F. Nikolay and M. Pesavento, “Learning directed-acyclic-graphs from large-scale double-knockout experiments,” in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 46–50.
- [NP17] —, “Learning directed-acyclic-graphs from multiple genomic data sources,” in *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2017, pp. 1877–1881.
- [NP18] —, “Learning dags using multiclass support vector machines,” in *2018 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2018, pp. 75–79.
- [NPKT17] F. Nikolay, M. Pesavento, G. Kritikos, and N. Typas, “Learning directed acyclic graphs from large-scale genomics data,” *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2017, no. 1, p. 10, Sep 2017.
- [NS12] P. Netrapalli and S. Sanghavi, “Learning the graph of epidemic cascades,” *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 211–222, Jun. 2012.
- [OFK⁺17] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, “Graph signal processing,” *Submitted to the Proceedings of the IEEE*, vol. 1, pp. 1–18, December 2017.
- [Pal05] M. Pal, “Random forest classifier for remote sensing classification,” *International Journal of Remote Sensing*, vol. 26, no. 1, pp. 217–222, 2005.
- [PR91] M. Padberg and G. Rinaldi, “A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems,” *SIAM Review*, vol. 33, pp. 60–100, 1991.
- [PW06] Y. Pochet and L. A. Wolsey, *Production Planning by Mixed Integer Programming*, ser. Springer Series in Operations Research and Financial Engineering. 233 Spring Street, New York, NY 10013, USA: Springer Science+Business Media, Inc., 2006.
- [Qui86] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar 1986.
- [RD10] J.-P. P. Richard and S. S. Dey, *The Group-Theoretic Approach in Mixed Integer Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 727–801.

- [RH10] J. Robinson and A. Hartemink, “Learning non-stationary dynamic bayesian networks,” *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3647–3680, 2010.
- [RK16] R. Rubinstein and D. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016, vol. 10.
- [Rut89] R. A. Rutenbar, “Simulated annealing algorithms: an overview,” *IEEE Circuits and Devices Magazine*, vol. 5, no. 1, pp. 19–26, Jan 1989.
- [Sal94] S. L. Salzberg, “C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993,” *Machine Learning*, vol. 16, no. 3, pp. 235–240, Sep 1994.
- [SBG17] Y. Shen, B. Baingana, and G. B. Giannakis, “Tensor decompositions for identifying directed graph topologies and tracking dynamic networks,” *IEEE Transactions on Signal Processing*, vol. 65, no. 14, pp. 3675–3687, July 2017.
- [Sch15] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85 – 117, 2015.
- [SKS⁺11] B. Szappanos, K. Kovacs, B. Szamecz, F. Honti, M. Costanzo, A. Baryshnikova, G. Gelius-Dietrich, M. J. Lercher, M. Jelasity, C. L. Myers, B. J. Andrews, C. Boone, S. G. Oliver, C. Pál, and B. Papp, “An integrated approach to characterize genetic interaction networks in yeast metabolism,” *Nature Genetics*, vol. 43, pp. 656–662, July 2011.
- [SL91] S. R. Safavian and D. Landgrebe, ““a survey of decision tree classifier methodology.” *ieee transactions on systems*,” vol. 21, pp. 660–674, 06 1991.
- [SM14] A. Sandryhaila and J. M. F. Moura, “Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,” *IEEE Signal Processing Magazine*, vol. 31, pp. 80–90, September 2014.
- [SMFR08] M. Schmidt, K. P. Murphy, G. Fung, and R. Rosales, “Structure learning in random fields for heart motion abnormality detection,” in *CVPR’08*, 2008.
- [SMMR17] S. Segarra, A. Marques, G. Mateos, and A. Ribeiro, “Network topology inference from spectral templates,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, 2017.
- [SRG08] I. Schizas, A. Ribeiro, and G. Giannakis, “Consensus inad hoc wsn with noisy links—part i: Distributed estimation of deterministic signals,” *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

- [SSBD14] S. Shalev-Schwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. 32 Avenue of the Americas, New York, NY 10013-2473, USA: Cambridge University Press, 2014.
- [SSMM17] R. Shafipour, S. Segarra, A. Marques, and G. Mateos, “Network topology inference from non-stationary graph signals,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5870–5874.
- [SSMM18] ———, “Topology inference of directed networks from diffused graph signals,” in *IEEE Data Science Workshop*. IEEE, 2018.
- [SWUM17] S. Segarra, Y. Wangt, C. Uhler, and A. Marques, “Joint inference of networks from stationary graph signals,” in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 975–979.
- [Tea18] A. H. Y. Tong and et al., “Global mapping of the yeast genetic interaction network,” *Science*, vol. 303, pp. 808–813, February 2018.
- [TSG17] P. Traganitis, Y. Shen, and G. Giannakis, “Topology inference of multi-layer networks,” in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2017, pp. 898–903.
- [Tur93] E. Turban, *Decision Support and Expert Systems: Management Support Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1993.
- [Vap00] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, USA: Springer-Verlag New York, 2000.
- [VP91] T. Verma and J. Pearl, “Equivalence and synthesis of causal models,” in *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, ser. UAI ’90. New York, NY, USA: Elsevier Science Inc., 1991, pp. 255–270. [Online]. Available: <http://dl.acm.org/citation.cfm?id=647233.719736>
- [WRL06] M. Wainwright, P. Ravikumar, and J. Lafferty, “Inferring graphical model structure using l-1-regularized pseudo-likelihood,” in *Proc. Neural Information Processing Systems Conference*, 2006.
- [WXL⁺14] J. Wang, X. Xu, D. Liu, Z. Sun, and Q. Chen, “Self-learning cruise control using kernel-based least squares policy iteration,” *IEEE Transactions on Control Systems Technology*, vol. 22, pp. 1078–1087, May 2014.
- [yea] “Sgd-saccharomyces genome database,” Website, <http://www.yeastgenome.org>.
- [YJ17] H. Yin and N. K. Jha, “A health decision support system for disease diagnosis based on wearable medical sensors and machine learning ensembles,” *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, pp. 228–241, October-December 2017.

- [YL07] M. Yuan and Y. Lin, “Model selection and estimation in the gaussian graphical model,” *Biometrika*, vol. 94, pp. 19–35, 2007.
- [ZLW10] S. Zhou, J. Lafferty, and L. Wasserman, “Time varying undirected graphs,” *Machine Learning*, vol. 80, no. 2-3, pp. 295–319, 2010.
- [ZLZZ16] S.-h. Zhan, J. Lin, Z.-j. Zhang, and Y.-w. Zhong, “List-based simulated annealing algorithm for traveling salesman problem,” *Computational Intelligence and Neuroscience*, vol. 2016, no. 8, March 2016.
- [ZRG16] L. Zhang, D. Romero, and G. Giannakis, “Fast convergent algorithms for multi-kernel regression,” in *2016 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2016, pp. 1–4.

List of Publications

Conference and Journal papers

1. F. Nikolay, M. Pesavento, G. Kritikos, and N. Typas, "Learning directed acyclic graphs from large-scale genomics data", *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2017:1
2. F. Nikolay and M. Pesavento, "Learning directed-acyclic-graphs from large-scale double-knockout experiments", in *24th European Signal Processing Conference (EUSIPCO)*, Budapest, 2016, pp. 46-50
3. F. Nikolay and M. Pesavento, "Learning directed-acyclic-graphs from multiple genomic data sources," *25th European Signal Processing Conference (EUSIPCO)*, Kos, 2017, pp. 1877-1881.
4. F. Nikolay and M. Pesavento, "Learning Dags using Multiclass Support Vector Machines," *IEEE Statistical Signal Processing Workshop (SSP)*, Freiburg im Breisgau, Germany, 2018, pp. 75-79.

List of Supervised Students

